TREBALL DE FI DE GRAU

Grau en Enginyeria en Tecnologies industrials

SIMULACIÓ DE CUES PER DIMENSIONAR PUNTS DE SERVEI

Memòria

Autora: Anna Cristóbal Ferré

Director: Pere Grima Cintas

Barcelona, Maig 2016

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Resum

El projecte final de grau *Simulació de cues per optimitzar punts de servei* exposat a continuació neix com a resposta a dues situacions.

Un primer objectiu és el de proporcionar una eina informàtica per a una primera avaluació estadística de determinats escenaris.

El programa informàtic, basat en llenguatge de programació Python, està conformat per una interfície gràfica d'interacció amb l'usuari, diferents mòduls de càlculs i finalment accés a la simulació, s'obre la qual es poden calcular i estudiar diferents aspectes d'interès personal. En cap cas el software pretén donar propostes d'optimització pels sistemes plantejats.

L'objectiu principal és doncs el de donar una alternativa per a la comparació de diferents sistemes de cues i poder així estudiar noves possibilitats permetent veure quina és la gestió dels servidors que millor s'adapta a les nostres necessitats.

Finalment, la creació d'interfícies gràfiques pels usuaris és un tema clau a l'hora de definir l'èxit d'un producte. Un disseny senzill, intuïtiu i accessible catapulta l'èxit d'un programa, mentre una interfície complexa i amb un grau elevat d'aprenentatge influeix negativament sobre el software. És per això que el següent projecte té com a objectiu secundari el de mantenir el programa accessible i amb una lògica de funcionament fàcil, permetent-ne l'ús a usuaris sense coneixements d'informàtica o estadística.

ÍNDEX

RESUM 1

ÍNDEX 3

1. INTRODUCCIÓ 5

2. OBJECTIUS I ABAST 6

2.1 OBJECTIUS 6

2.2 ABAST DEL PROJECTE 6

3. ANÀLISI D'ALTRES SOFTWARE 9

4. EINES UTILITZADES 11

4.1 PYTHON 11

4.1.1 XWLT 11

4.1.2. PYQT 11

4.2 QT DESIGNER 12

4.3 ESTUDI D'ALTERNATIVES 12

5. SIMULACIÓ 14

5. 1 SISTEMES DE CUES 14

5.1.1 Conceptes bàsics dels sistemes 14

5.2. DISTRIBUCIONS ESTADÍSTIQUES 15

Patró d'arribada dels clients: 15

Patró de servei dels servidors 16

5.3 SIMULACIÓ 17

MODEL 17

MODELITZACIÓ 18

SIMULACIÓ 18

5.4 RESULTATS 19

6. FUNCIONS I ESTRUCTURA DEL SOFTWARE 21

6.1 ESTRUCTURA DEL SOFTWARE 21

6.2 UTILITAT DEL SOFTWARE 22

PARÀMETRES I VARIABLES D'ENTRADA: 22

SISTEMA TIPUS 1: Un tipus de servei – Una única cua. 26

SISTEMA TIPUS 2: Multiservei – Una única cua 28

SISTEMA TIPUS 3: Un tipus de servei – Una cua per servidor 29
SISTEMA TIPUS 4: Multiservei – Una cua per servidor 30

7. DISSENY I FUNCIONAMENT DE LA INTERFÍCIE 32
7.1 PANTALLA INICI 32
7.2 PANTALLES INTERMITGES..... 33
7.3 PANTALLA RESULTATS 37

8. PRESENTACIÓ DE RESULTATS 38
SISTEMA TIPUS 1: Un tipus de servei – Una única cua. 39
SISTEMA TIPUS 2: Multiservei – Una única cua 40
SISTEMA TIPUS 3: Un tipus de servei – Una cua per servidor 41
SISTEMA TIPUS 4: Multiservei – Una cua per servidor 41

9. EXEMPLE D'APLICACIÓ 43

10. MILLORES I FUTUR DEL PROJECTE 45

11. VALORACIÓ IMPACTE AMBIENTAL 47

12. VALORACIÓ IMPACTE ECONÒMIC 48

CONCLUSIONS 49

AGRAÏMENTS 51

BIBLIOGRAFIA 53
Referències bibliogràfiques..... 53
Bibliografia complementària..... 54

1. Introducció

La sensació d'estar perdent temps en una cua d'espera és una situació comuna per a tots. Les cues ens semblen una etapa natural de molts processos, i tot i que esperem en situacions quotidianes: un peatge, un lavabo, per facturar les maletes a l'aeroport o fins i tot perquè ens atengui un operador, en general, com a clients no ens agrada esperar. Per què cal esperar? Quant temps m'he d'esperar?

La resposta és senzilla, doncs està clar que en algun moment la capacitat del servei és menor a la demanda. Aquestes limitacions es poden compensar invertint en elements que augmentin la capacitat però els recursos són sempre limitats i cal que donin resposta als diferents requeriments de les organitzacions. En funció d'això s'estableix qui o què espera i per això a vegades, les cues són bones. Ens permeten veure la importància i qualitat del producte que anem a comprar, o del servei que anem a rebre i reflexionar així en els nostres requeriments.

En qualsevol cas, en la majoria d'àmbits de la societat es contempla un procés de millora continua cap a l'excel·lència i això és el que ha incitat aquest projecte. Desenvolupar una eina que permeti respondre a si l'organització establerta és la millor gestió dels recursos i permeti avaluar noves estratègies, dins els límits possibles, per veure'n l'efecte abans d'invertir-hi.

2. OBJECTIUS I ABAST

2.1 OBJECTIUS

L'objectiu del projecte és el disseny d'un software que permeti extreure conclusions estadístiques i patrons de comportament un cop determinades les característiques del sistema.

Aquest software cal que contempli tres aspectes importants:

- En primer lloc, una interfície gràfica capaç d'interaccionar amb usuaris sense cap noció de programació o informàtica.
- En segon lloc, una llibreria de funcions que permeti a altres interessats adaptar o desenvolupar diferents aplicacions de manera ràpida i senzilla prenen les generades de base.
- En tercer lloc, una aplicació que serveixi per comparar diferents models de sistema i sigui capaç de dimensionar els servidors en funció del temps d'espera.

Finalment i a nivell personal, el projecte té com a objectius complementaris:

- Adquirir coneixements de simulació i modelització de processos.
- Recuperar i ampliar els coneixements apresos en l'assignatura d'informàtica sobre programació amb llenguatge d'alt nivell Python.
- Adquirir coneixements de programació d'interfícies gràfiques.

2.2 ABAST DEL PROJECTE

La teoria que hi ha al darrera de la formació de cues és sovint massa complexa per permetre modelar matemàticament totes les situacions mundialment possibles. Els recursos alternatius a l'estudi analític consisteixen principalment en simulacions en ordinador i/o en l'anàlisi de dades obtingudes experimentalment. El projecte present es

basa precisament en la simulació per intentar modelitzar sistemes i així poder-ne estudiar el funcionament o predir el comportament.

Cal tenir present que la simulació és la imitació d'alguna cosa real i en aquesta línia s'ha procurat incloure dins les funcionalitats del software el màxim nombre d'escenaris possibles. Tot i així, el ventall de possibilitats és infinit i per això es detallen a continuació algunes suposicions prèvies dels casos contemplats:

- Cap client abandona la cua.
- Els clients no es poden canviar de cua.
- Els processos són únics. És a dir, no es contemplen encadenaments o relacions de servidors o punts de servei, com podria passar en l'Agència Tributària o a la Prefectura Provincial de Trànsit, on al primer taulell s'informa del taulell definitiu on t'atendran.
- No hi ha clients que puguin passar a la cua per davant de clients anteriors.

Al capítol 5 d'aquesta mateixa memòria s'explica amb més detall els casos contemplats i la lògica utilitzada pel seu desenvolupament. De la mateixa manera al capítol 7 es plantegen millores i noves funcionalitats que es podrien incloure per tal d'oferir al programa més versatilitat.

Per últim, destacar que la finalitat del projecte no és arribar a conclusions estadístiques a partir de les dades, sinó oferir les bases necessàries per tal que l'usuari pugui arribar a elles.

3. ANÀLISI D'ALTRES SOFTWARE

Abans de començar amb el plantejament del projecte i per tal que aquest ofereixi noves opcions als usuaris, i encaminar així el software desenvolupat a omplir un buit dins el món de la simulació de cues, s'ha realitzat una anàlisi d'eines alternatives disponibles i les seves característiques.

A continuació es mostra la taula comparativa:

SOFTWARE	***ACCESSIBILITAT***	***ENTORN TÈCNIC***	***COMPLEXITAT***	***CARACTERÍSTIQUES***	***FONT***
PowerDEVS	Software Lliure	Windows i Linux, implementat amb C++	Mitja	Simulació d'esdeveniments discrets i continus enfocats a la logística.	[1]
Graphical SpreedSheat Queuing Simulation	Taules d'Excel Online	Macros d'Excel	Bàsica	Simulació de cues amb usuaris que abandonen la cua. Cues úniques i fins a dotze servidors.	[2]
JaamSim i altres.	Open-source	Windows, implementat amb Java	Mitja	Simulació d'esdeveniments discrets i continus enfocats a la logística.	[3]
Arena, Witness, Automod...	Pagament	Windows	Alta	Esdeveniments discrets i continus, ús de flowcharts i workflows.	[4]

Taula 1. Comparativa d'altres softwares

Com a conclusió de l'anàlisi és busca un software de complexitat bàsica, implementat amb Python per tal que es pugui utilitzar als sistemes informàtics: Windows, Linux, OS X.

4. EINES UTILITZADES

A continuació es detalles les eines utilitzades tan pel desenvolupament del software com pel seu funcionament.

4.1 PYTHON

El programa informàtic està basat en el llenguatge de programació Python.

Aquest és un llenguatge d'alt nivell orientat a objectes. A efectes pràctics això significa que és un tipus de llenguatge més proper a com expressaríem les coses a la vida real, més intuïtiu, i per tan té un codi més llegible i una sintaxi més curta.

Entre diferents avantatges d'aquest llenguatge cal remarcar que forma part del programa de software lliure GNU General Public License, és a dir, permet la utilització, còpia i modificació del software. Això es tradueix en que existeixen una gran varietat de llibreries per desenvolupar amb molta diversitat de funcions i temàtiques. Per aclarir conceptes, les llibreries són paquets de software que contenen funcions per facilitar el disseny del software. En l'apartat següent es detallen les llibreries utilitzades per la realització d'aquest projecte.

D'altra banda és important tenir present que existeix una comunitat molt activa a la xarxa que facilita la resolució de dubtes i webs amb documentació.

Finalment, remarcar que és u llenguatge àmpliament utilitzat en grans empreses del sector tecnològic com Google o Dropbox i per tan, un coneixement d'aquest presenta una millora notable en el perfil professional si es vol dedicar-se al sector.

4.1.1 XWLT

Llibreria destinada a crear i modificar arxius Excel des de Python. El seu ús és necessari en aquest projecte per la presentació de resultats.

4.1.2. PYQT

És una llibreria pensada pel disseny i programació de interfícies gràfiques. Conté múltiples opcions per representar diferents pantalles, botons i altres elements que es vulgui afegir a

l'aplicació. Conté de la mateixa manera, funcions que permeten que el programa pugui interaccionar amb l'usuari per exemple quan aquest introdueix un valor o prem un botó.

Aquesta llibreria s'ha utilitzat, junt amb el programa Qt Designer, pel disseny de la interfície i programació de les interaccion dels programa.

4.2 QT DESIGNER

És un programa del projecte Qt pel disseny d'interfícies d'usuari. Permet doncs establir el disseny, geometria, localització...dels elements que interactuen amb l'usuari.

Aquest programa facilita el disseny, especialment la part de disposició dels elements, tot i així no es prou complet ni per dissenyar tota la interfície ni per programa-la, és necessari doncs l'ús de la llibreria PyQt per completar el codi.

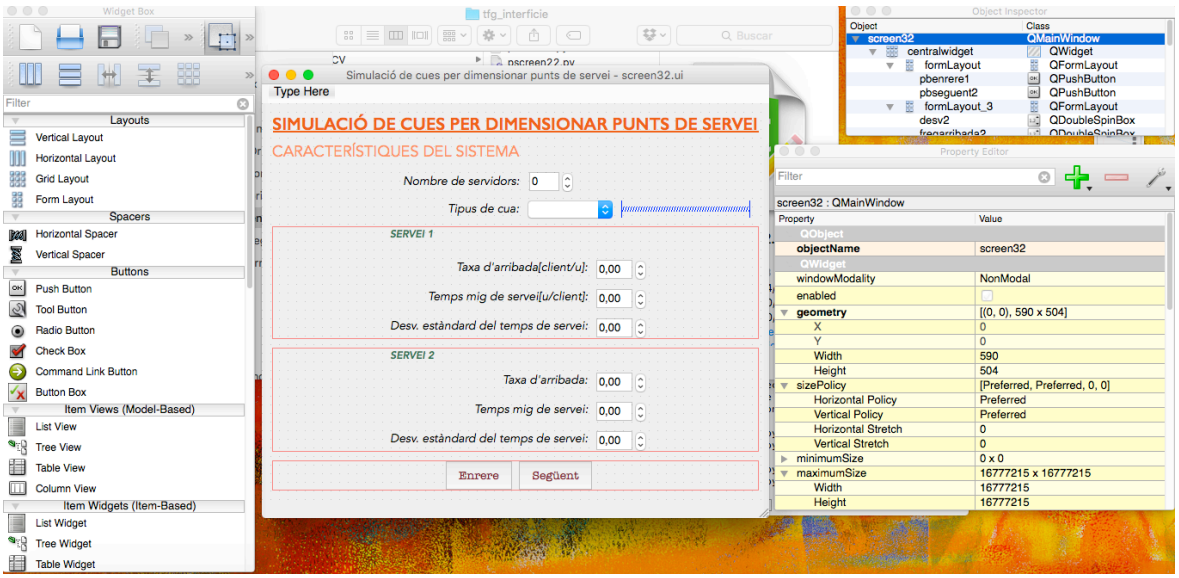


Fig 1. Disseny de la interfície amb Qt Designer

4.3 ESTUDI D'ALTERNATIVES

En un primer moment el projecte s'havia plantejat per desenvolupar-se en un entorn d'Excel. El programa i la interacció amb l'usuari es pensaven compilar amb llenguatge Visual Basic. Aquest és un llenguatge de programació dirigit per esdeveniments, i per tant la sintaxi és inflexible i poc intuïtiva. D'altra banda només genera executables per Windows i només

Microsoft pot decidir-ne l'evolució. Finalment els programes generats són relativament lents i grans en comparació amb els generats per Python.

Tenint en compte les característiques d'aquest llenguatge de programació i tot que la idea de desenvolupar el programa en un entorn Windows era viable en el sentit de proximitat i comoditat per l'usuari, es va decidir desenvolupar el software amb Python. Això ha permès més versatilitat, opció de millora i implementació més ràpida i fàcil o poder utilitzar l'aplicació en altres sistemes operatius com OS X o Linux.

De totes formes i per una major utilitat, les dades generades en la simulació és presenten en format Excel permeten així una visualització clara dels resultats i el posterior anàlisi.

5. SIMULACIÓ

Aquest capítol es crea amb la intenció de presentar els fonaments teòrics necessaris pel desenvolupament del projecte. Per això es procura incloure les següents àrees d'estudi: conceptes relacionats amb els sistemes de cues, distribucions estadístiques i modelització.

5. 1 SISTEMES DE CUES

Un sistema de cues es pot descriure com un conjunt de clients que arriben al sistema buscant un servei, si el servei no és immediat esperen, i finalment surten del sistema un cop han sigut atesos.

El terme "client" s'utilitza com a generalització doncs no implica un ser humà. Podria tractar-se d'una llista de documents esperant a ser impresos o peces que cal processar. De totes formes, i per a major comprensió, s'utilitzarà el terme client per referir-se al usuari del sistema en qüestió.

Fig 2 :Sistema de cua bàsic

5.1.1 Conceptes bàsics dels sistemes

El terme sistema s'utilitza en diferents contextos i és per això que és fa difícil trobar una definició concreta però segons Law y Kelton (2000) [5] es defineix; "un sistema és una col·lecció d'entitats com per exemple persones o màquines, que actuen i interaccionen entre sí, per tal de complir un objectiu"

En tot cas perquè un sistema pugui ser, cal que estigui format per diferents parts i que aquests interaccionin entre ells.

Entitat: Segons la Universitat de Buenos Aires http://materias.fi.uba.ar (2005) [6] , una entitat és alguna cosa (objecte), que té realitat física o objectiva i que posseeix un conjunt de característiques (atributs) que el fan distingible de la resta, per exemple els clients o els serveis que s'ofereixen.

Atributs: És el conjunt de propietats que caracteritzen les entitats del sistema, per exemple: el temps d'arribada, el temps de servei o el número de servidors.

Estat: Tal com estableix Barceló(1996) [7] són els valors i/o condicions dels atributs en un instant donat, de forma que es pot saber si una acció és pot executar. Per exemple, el nombre de clients a la cua, el temps de simulació o si un servidor està lliure o ocupat.

Esdeveniment : Són els diferents fets que passen en instants de temps i donen lloc als canvis dels estats al sistema. Per exemple: una arribada, o sortida d'una entitat.

5.2. DISTRIBUCIONS ESTADÍSTIQUES

Per tal de modelar les cues es pren en especial consideració dues característiques bàsiques de les cues: el patró d'arribades i el patró de servei dels servidors. L'anàlisi de les cues està determinat en gran part per la distribució dels temps d'arribada i de servei.

A la vida real, i tenint en compte els infinits casos possibles, aquestes distribucions poden prendre qualsevol forma, mentre no siguin negatives, ara bé, per tal de realitzar la simulació cal especificar la suposada forma de cada distribució i perquè la simulació resulti útil, cal que la distribució utilitzada sigui prou realista.

Patró d'arribada dels clients:

En situació de cues habituals l'arribada és estocàstica, es a dir que depèn d'una variable aleatòria. En aquest cas es suposa per tan que l'arribada d'un nou client esdevé de forma independent i que el temps entre arribades segueix doncs una distribució exponencial, o el que és el mateix, que el ritme d'arribades, client per unitat de temps (u.t.), segueix una distribució de Poisson.

Al programa presentat l'usuari té l'opció d'indicar la taxa d'arribades per a cada tipus diferent de servei que plantegi. Així doncs tenint present la Llei exponencial amb la funció de densitat de probabilitat següent:

$$f(t)=\lambda \exp(-\lambda t)$$

per generar una nova mostra de temps entre arribades, s'utilitza:

$$x = \frac{-1}{\lambda} \ln y \qquad (1)$$

De l'equació (1), el paràmetre λ és la taxa d'arribades indicada per l'usuari (clients/u.t.). D'altra banda el programa genera un número aleatori utilitzant la funció *"random()"* de la llibreria *Random* de Python [8], que retorna un número entre 0 i 1 i així la variable x ens indica quin serà el nou temps entre arribades.

Cal tenir en compte que s'ha de calcular un nou temps entre arribades per a cada client.

Patró de servei dels servidors

Pel que fa al temps de servei, aquest pot fàcilment no seguir una distribució exponencial. De fet, els temps de servei seguiran una llei exponencial quan el temps de servei sigui aleatori i independent, per exemple en un mostrador d'un punt d'informació, on el client pot demanar coses molt diferents i el temps de servei per tant, variar molt entre un client o un altre. És comú però que el temps de servei sigui constant si el servidor sempre realitza la mateixa seqüència d'operacions, i llavors els temps de servei tendeixen a aproximar-se al temps esperat de servei. En aquest cas, on les petites variacions vindrien donades per l'eficiència del servei, la distribució exponencial no seria una bona aproximació a les distribucions del temps de servei.

És per això que en aquest cas, el programa permet a l'usuari escollir quina distribució vol utilitzar pel temps de servei:

- Distribució Exponencial:

En aquest cas l'usuari introdueix la mitja del temps de servei μ (u.t/client). Igual que amb el temps d'arribada, el programa calcula un nou temps de servei. Aquest cop però:

$$x = -\mu \ln y \qquad (2)$$

- Distribució Normal truncada als valors positius:

L'usuari ha d'indicar la mitja del servei (μ) i la desviació estàndard (σ) i el programa mitjançant la llibreria *random()* utilitza la funció *random.normalvariate(mu, sigma)* que donant els dos paràmetres ja retorna un nou temps de servei.

- Distribució Uniforme:

El programa permet indicar un mínim i un màxim de temps de servei i, utilitzant la funció *random.uniform(a, b)* de la llibreria *random()*, el programa calcula un nou temps de servei aleatori dins l'interval indicat.

5.3 SIMULACIÓ

L'ús de la simulació és útil per tal de predir el comportament i avaluar així decisions sobre el nostre sistema. La simulació és doncs, una eina per experimentar a través d'un model matemàtic del sistema permetent així l'estalvi d'inversions precipitades o incertes.

MODEL

Els components del sistema vists poden ser representats de forma matemàtica gràcies a diferents models amb l'objectiu d'estudiar el comportament del sistema. Donat que el camp

de la modelització és molt ampli, a continuació es detalla el tipus de model escollit per les nostres simulacions[9]:

Model Discrets per Esdeveniments: Tipus de model dinàmic, estocàstic i discret on les variables d'estat canvien de valor en instants no periòdics de temps, sense estar dirigides per un rellotge. Aquests instants de temps es corresponen per exemple a l'arribada d'un nou client.

MODELITZACIÓ

Per tal de modelitzar el sistema cal primer recollir les dades, per fer-ho el programa permet introduir les dades següents:

- El temps de simulació: Limitarà el rellotge del model indicant el final de la simulació.
- El tipus de servidors i de cues que té el sistema: Configuracions dels diferents sistemes possibles que s'expliquen amb més detall al capítol 6 d'aquesta memòria.
- El nombre de serveis independents que s'ofereixen al sistema: El programa permet introduir fins a 3 serveis diferents en cada estudi.
- El nombre de servidors que hi ha: Servidors independents per a cada servei o servidors del tipus multiservei, explicat amb més detall al capítol 6.
- Paràmetres del model i distribucions de probabilitat de les seves variables aleatòries per a cada servei: Com s'ha comentat, el sistema permet escollir entre diferents opcions pel que fa a les distribucions dels temps de servei.

Al capítol 7 es fa una explicació detallada de com l'usuari pot introduir aquestes dades.

SIMULACIÓ

En aquest apartat es procura fer una aproximació a l'estructura del codi utilitzada per generar les dades de la simulació per a una millor comprensió del lector.

En general els codis de simulació conten amb les mateixes tres parts:

1.- Inici de la simulació:

- Determinació de dades anteriorment esmentades del sistema.

- Inicialització de les variables del sistema.

- Generador de les mostres de variables aleatòries: Generació dels temps entre arribades i temps de servei per a cada client.

- Determinar l'esdeveniment que inicia la simulació.

2.- Iteració:

- Funcions que processen l'esdeveniment i actualitzen l'estat dels elements del sistema.

- Emmagatzemar les noves dades creades.

- Determinar el proper esdeveniment.

- Actualitzar el rellotge.

- Actualitzar els comptadors estadístics.

3.- Fi

- Donar els resultats de la simulació.

5.4 RESULTATS

Com s'ha comentat anteriorment el programa està pensat per a usuaris que no tenen coneixement de programació ni estadística, així que els resultats cal que es presentin de forma visualment clara. De totes formes, i pensant en altres usuaris possibles, és important presentar-los de forma útil per un posterior anàlisi si es desitja. Amb aquestes consideracions, el format de presentació de resultats és un full Excel. Al capítol 8

d'aquesta memòria es fa una explicació detallada de com interpretar les taules obtingudes amb el programa.

6. FUNCIONS I ESTRUCTURA DEL SOFTWARE

En aquest capítol s'explica en detall la lògica de l'aplicació i el raonament utilitzat per a la seva programació. El projecte contempla quatre escenaris diferents i a continuació es procura descriure l'estructura general del programa informàtic i l'estructura detallada de cada cas.

6.1 ESTRUCTURA DEL SOFTWARE

Els diferents aspectes del software queden agrupats en varis fitxers, cada un amb una missió concreta.

En un primer lloc trobem l'arxiu *"principalpral.py"*, aquest s'encarrega d'executar el programa, obrir els fitxers de càlculs per tal de realitzar la simulació i establir les relacions entre pantalles.

D'altra banda tenim els fitxers que s'encarreguen del disseny de les pantalles de la interfície gràfica i de emmagatzemar les característiques i paràmetres que l'usuari introdueix per la seva simulació. El programa conta amb vint pantalles diferents, per cobrir totes les opcions, els fitxers en qüestió s'anomenen: *"pscreen....py",* i el número de pantalla corresponen. Tenir aquests fitxers per separat permet de forma ràpida i fàcil accedir al disseny de la pantalla en cas que es vulgui o calgui realitzar canvies en l'organització dels elements, o en els títols utilitzats, així com si es volen afegir nous elements per tal que l'usuari pugui indicar nous paràmetres que es puguin necessitar en un futur.

A continuació entra en joc l'arxiu de *"càlculs.py",* i és aquest el que conté la simulació pròpiament dita. En aquest document s'hi troben quatre funcions, cada una per simular un dels quatre casos possibles.

Finalment, el fitxer *"programappl.py"* és l'encarregat de llegir els resultats de la simulació per tal de tractar les dades creades en l'arxiu de "*calculs.py"* i poder així presentar-les en un format còmode i pràctic (Excel) per la seva visualització, comprensió i futur estudi.

Els fitxer de "programapral.py" i "càlculs.py" estan disponible a l'annex d'aquesta memòria. Els altres fitxers no s'han afegit per la seva extensió.

6.2 UTILITAT DEL SOFTWARE

A la pràctica el programa presenta dues funcions bàsiques, La primera simular el cas introduït a nivell informatiu, i la segona, generar una solució que determinarà el nombre de servidors o punts de servei mínim que cal que tinguem per no superar el temps d'espera màxim desitjat que haguem introduït.

PARÀMETRES I VARIABLES D'ENTRADA:

A l'iniciar el software, aquest ens demana a través de dues pantalles que caracteritzem en la primera pantalla el sistema, i en la segona el o els serveis, fins a un màxim de tres.

Els elements que es demanen per tal de generar la simulació i la solució són els següents:

CARACTERÍSTIQUES DEL SISTEMA:

- *Número de serveis diferents que hi ha:* Aquí cal indicar el nombre de serveis diferents que ofereix el nostre sistema fins un màxim de tres. O es poden posar diferents serveis independents per generar les simulacions a la vegada i així poder-los comparar sense haver de generar dues simulacions diferents.

- *Temps de simulació:* Nombre d'unitats de temps durant el qual volem generar les simulacions. Això pot fer referència a la franja que l'establiment està obert o una franja més concreta, per exemple en hora punta, per poder visualitzar el cas on hi ha més afluència de gent.

- *Tipus de servidor:* Entenen per servidor, els punts de servei, ja siguin taquilles, caixers, lavabos... En aquesta classificació les dues possibilitats que es poden escollir són: *Un tipus de servei*", o *"Multiservei".*

  "Un tipus de servei" implica que cada servidor pot atendre només un tipus de client. Per exemple, els lavabos exclusius per a un gènere (homes/ dones), la línia de peatge exclusiva per a *Teletac* o la cua de facturació a l'aeroport exclusiva per a usuaris amb tarifa *Premium.*

Fig 3. Cada servidor atén un tipus de servei

D'altra banda, l'opció de servidors "*Multiservei"* dóna lloc a una simulació on els servidors poden atendre el 100% dels clients. És a dir, qualsevol servei o producte es pot gestionar des de qualsevol punt de servei, mostrador... Aquest es també un cas típic en molts escenaris quotidians com ara és el cas dels lavabos mixtes, les cues del supermercat, els punts d'informació d'establiments (on es pot preguntar informació de molts tipus, o demanar per objectes perduts...), o el pas de seguretat de l'aeroport, doncs tothom passa pel mateix sense importar les característiques personals.

Fig 4. Cada servidor atén tots els tipus de servei

- *Tipus de cua*: Aquesta divisió es fa entre "*Una única cua"* o *"Una cua per a cada servidor"*.

Amb l'opció *"Una única cua"* s'entén que hi ha una sola cua per a cada grup de servidors. És a dir, si s'ha marcat l'opció de "*Un tipus de servei"* i s'ha indicat que aquest servei té n servidors, amb aquesta opció establirem que a aquests n servidors els arriben clients que esperen en una mateixa cua, seguin una disciplina de cua del tipus FIFO (*First in Fist out),* és a dir, el primer que arriba, és el primer en ser atès. Aquest és un sistema que veiem per exemple en la cua del cine, doncs l'establiment té n taquilles i tots els espectadors esperen en una sola cua de tal manera que, quan una taquilla queda buida, el primer de la cua l'ocupa i pot ser atès.

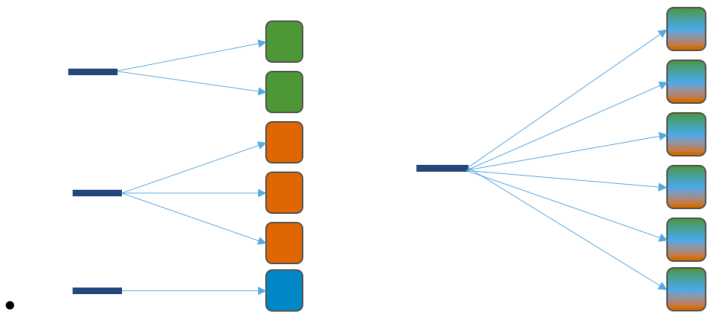


Fig 5. Generació de cues úniques en funció del tipus i nombre de servidors i del nombre de serveis que es volen simular simultàniament.

La segona opció pel que fa a les cues, "*Una cua per a cada servidor",* estableix un sistema on com indica el nom cada punt de servei té la seva pròpia cua. Aquesta logística es segueix en situacions com les dels peatges, doncs cada línia de pas té la seva pròpia cua de vehicles, o generalment per l'ús dels *PolyKlin* (vàters portàtils) utilitzats en festivals de música i altres esdeveniments com fires o festes públiques al carrer, ja que cada un acostuma a tenir la seva pròpia cua davant del mateix.

Aquesta tipus d'ordre segueix una disciplina diferent a l'anterior, doncs els clients es col·loquen en les diferents cues segons la seva longitud. És a dir, per crear aquest tipus d'escenari s'ha establert la lògica de que quan un client arriba observa quina és la cua més curta i s'hi afegeix. Això és discutible doncs evidentment no sempre els clients segueixen aquest comportament, però de totes formes és el més comú i pràctic i s'ha agafat com a cas general. Ara bé, tots hem experimentat la situació de ficar-nos a una cua perquè és més curta i que la del costat avanci més ràpida, tot i així i com s'ha comentat a l'abast del projecte, el programa no contempla l'opció de canviar-se de cua. Així doncs, en aquest escenari, els clients no responen a la disciplina FIFO ja que es possible que un client sigui atès més tard que un client amb arribada al sistema posterior, depenen de la cua on es situï cadascú.

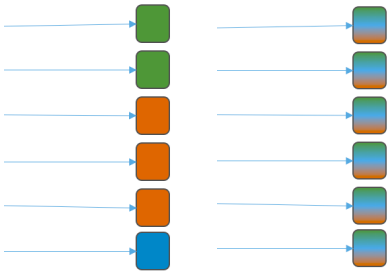


Fig 6. Generació de cues amb l'opció una cua per a cada servidor.

Depenen de les configuracions, el programa permet doncs generar quatre tipus d'escenaris diferents. Les diferents combinacions i les relacions entre els sistemes es mostren en la figura 7. Els següents apartats expliquen amb més detall les particularitats de cada un.

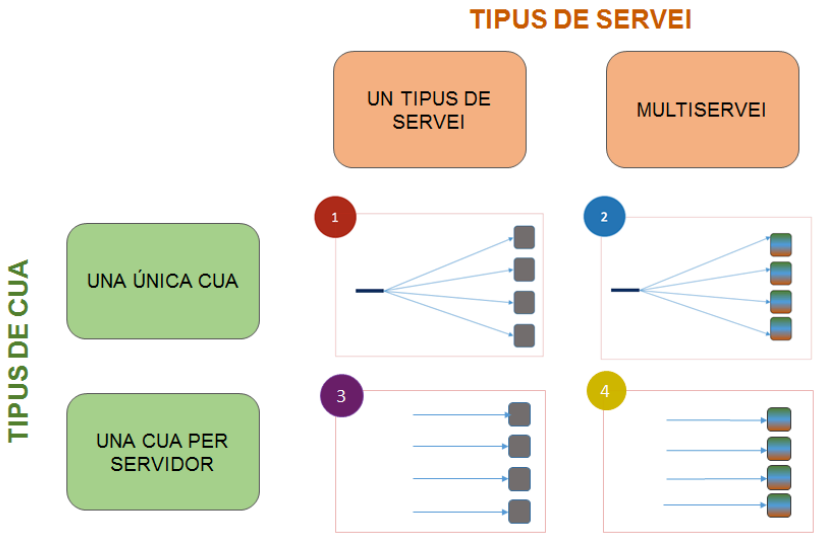


Fig 7. Taula dels quatre escenaris possibles a simular.

• *Distribució del temps de servei:* Aquesta opció serveix, per com s'ha esmentat anteriorment poder escollir si el temps de servei segueix una distribució Normal, Exponencial o Uniforme. Segons l'opció indicada, el programa ens demanarà introduir els paràmetres corresponents per tal de generar les mostres.

CARACTERÍSTIQUES DELS SERVEIS

• *Número de servidors:* Aquí caldrà indicar quants n'hi ha. Si s'ha indicat que els servidors són destinats a atendre un únic servei caldrà indicar quants servidors té cada servei per separat. Si pel contrari s'ha indicat que els servidors són multiservei, la pantalla ens donarà l'opció d'indicar per a tots els serveis agrupats quants servidors hi ha.

• *Temps d'espera màxim desitjat:* Aquest és el paràmetre clau i en funció del qual el programa genera una solució. Com hem dit, el programa genera una simulació (informativa) del sistema presentat, i una solució segons el temps màxim que es vol d'espera.

En el cas de la simulació informativa, el programa simplement indica, tenint en compte tots els paràmetres que s'han introduït, quin és el temps màxim d'espera (que pot superar el desitjat, i quants clients tenen un temps major al desitjat.

D'altra banda, la solució que genera el programa, en un arxiu Excel diferent, indica el nombre de servidors mínim que calen per tal que cap client hagi d'esperar un temps d'espera superior al indicat.

Cal tenir present però, que les dades que utilitza el programa pel temps d'arribada i pel temps de servei, s'han generat a partir de funcions de distribució que contenen variables aleatòries, i que per tant, la resposta no és 100% fiable. Doncs caldria generar varies simulacions per tenir una mostra suficientment gran abans de prendre una decisió.

- *Taxa d'arribada:* Freqüència d'arribada dels clients segons el servei. Indicat amb les unitats: client/unitat de temps.

- *Temps mig de servei, desviació estàndard, màxim temps de servei o mínim temps de servei:*

  Són els diferents paràmetres que cal introduir segons la distribució que s'hagi escollit pel temps de servei

A continuació s'explica amb més detall els quatre escenaris possibles segons la configuració triada.

SISTEMA TIPUS 1: Un tipus de servei – Una única cua.

Aquest sistema respon a situacions on tenim n servidors que atenen un únic servei i a més, es forma una única cua. Es dóna aquesta situació per exemple, a les taquilles del cine o als lavabos, només si aquests no són mixtos. Per aclarir l'esquema es presenta la figura següent:

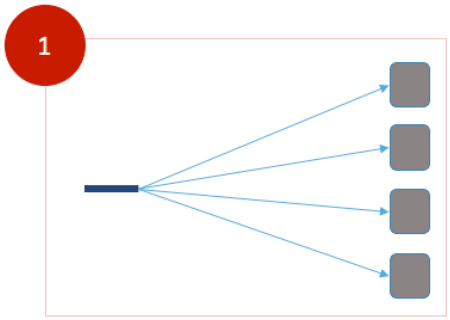


Fig 8. Representació esquemàtica de la configuració de servidors que atenen un únic servei i clients que esperen en una única cua.

En el cas que s'introdueixin al programa diferents serveis, aquest els tractarà de forma independent, mostrant-los cada un en un full del llibre d'Excel diferent.

Pel que fa al codi, aquest tipus de sistema segueix la lògica següent:

Per a cada servei:

1.- Es determinen els paràmetres del servei: número de servidors, taxa d'arribada, paràmetres del temps de servei i distribució estadística.

2.- Es genera una primera arribada que iniciarà la simulació.

3.- El nou client mira si hi ha servidors lliures, si és així es col·loca al primer servidor lliure.

4.- Si tots els servidors estan ocupats el client es situa a la cua.

5.- Es genera una nova arribada actualitzant el rellotge.

6.- S'actualitzen els estats dels servidors.

Per a cada client es calculen una sèrie de dades que després seran plasmades als fulls de resultats per facilitar-ne l'anàlisi estadístic. Aquestes dades es detallen al capítol 8.

Al final de la simulació el programa retorna per a cada servei:

- Una llista de clients
- Una llista per a cada servidor del servei on s'indica quins clients atén.

SISTEMA TIPUS 2: Multiservei – Una única cua

Sistemes del tipus lavabo mix, on cada usuari es caracteritzat per uns paràmetres de servei diferent (dones, homes…) però tot i així utilitzen el mateix servidor, vàter en aquest cas. Però podria aplicar-se als caixers automàtics d'una sucursal, on cada client hi acut per serveis diversos però utilitzen el mateix caixer, esperant en una sola cua a que la persona de davant acabi.

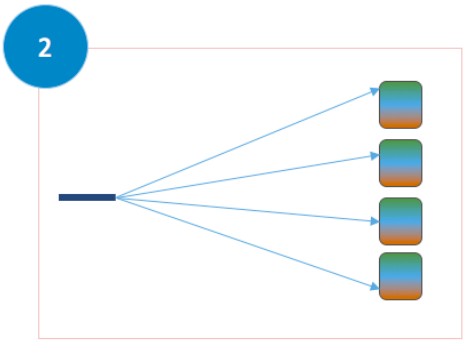


Fig 9. Representació esquemàtica de la configuració de servidors multiservei amb una única cua.

Pel que fa a la lògica del codi:

Per simular aquest tipus de sistema i donat que els diferents tipus de serveis interactuen entre sí, es suposa per generar les arribades que la població de cada servei és independent a les altres i per tan, per a cada servei que s'hagi indicat des de les pantalles del programa, i fins que no es superi el temps de simulació estipulat per l'usuari:

1.- Es determinen els paràmetres del servei: número de servidors, taxa d'arribada, paràmetres del temps de servei i distribució estadística.

2.- Es generen totes les arribades possibles dins el temps de simulació establert i es guarden en una llista ordenada per temps d'arribada.

Al final s'obté una llista per servei que conté les arribades per a cada servei, on s'indica el temps d'arribada del client i el temps de servei. Durant la iteració de la simulació aquestes llistes es filtren i s'ordenen segons els temps d'arribada per generar una sola cua definitiva. És per això, que en les taules generades per aquest tipus de simulació, el temps que marca el rellotge quan arriba a un client no sempre respon al temps del rellotge anterior més el temps entre arribada generat per la funció distribució, doncs aquest temps entre arribades és el que hi ha entre clients del mateix servei.

Iteració:

3.- Es busca d'entre les llistes de cada servei el següent client, que serà el que tingui el temps d'arribada més petit.

4.- Aquest mira l'estat dels servidors, si algun està lliure s'hi col·loca i sinó és col·loca a la cua.

5.- Es busca el següent client d'entre les llistes d'arribades generades (aquell que tingui el temps d'arribada més petit).

6.- S'actualitzen els estats dels servidors i de la cua.

Al final de la simulació el programa retorna una única llista de clients, doncs els servidors són multiservei i la cua és comuna. I una llista on s'indica en quin mostrador s'ha atès cada client.

SISTEMA TIPUS 3: Un tipus de servei – Una cua per servidor

En general, els sistemes on hi ha una cua per servidor tenen temps d'espera més elevats. Això es deu a que, com s'ha comentat, els clients es col·loquen en funció de la longitud de la cua desconeixent el temps de servei que comportarà atendre cada client. Tot i així, aquest tipus de configuració és la seguida en el peatge per exemple, on a vegades, la cua del costat m'ha més de pressa que la nostra tot i ser del mateix tipus (manual, automàtica...).

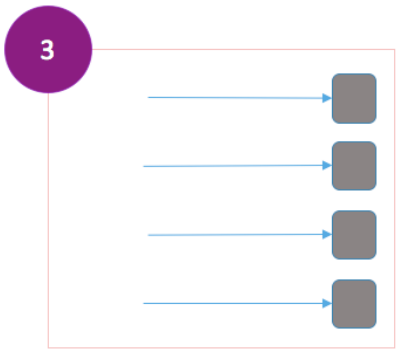


Fig 10. Representació esquemàtica de la configuració de servidors que atenen un únic tipus de client i amb una cua per servidor.

Pel que fa a la lògica del codi:

Per a cada servei:

1.- Es determinen els paràmetres del servei: número de servidors, taxa d'arribada, paràmetres del temps de servei i distribució estadística.

2.- El client mira si hi ha algun servidor lliure, si és així s'hi col·loca, sinó, mira quina és la cua de menys longitud i s'hi col·loca.

3.- Es genera una nova arribada actualitzant el rellotge.

4.- S'actualitzen els estats dels servidors i de la cua.

Aquest tipus de sistema és el que més cues genera. Doncs el programa genera una cua per servidor i servei.

SISTEMA TIPUS 4: Multiservei – Una cua per servidor

Finalment, un tipus de sistema on cada punt de servei pugui atendre el 100% dels clients i tingui la seva pròpia cua. És l'exemple dels supermercats o les màquines de *vending* del metro, o de la cafeteria mateix de l'ETSEIB.

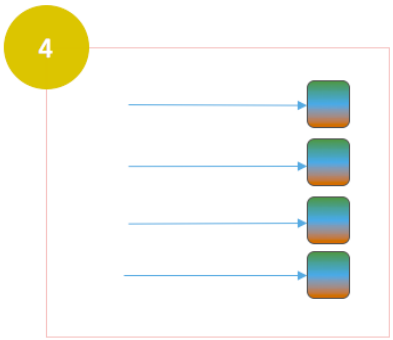


Fig 11. Representació esquemàtica de la configuració de servidors multiservei i amb una cua per servidor.

Pel que fa a la lògica del codi:

Per simular aquest tipus de sistema i donat que els diferents tipus de serveis interactuen entre sí,, es suposa per generar les arribades el mateix sistema que al cas 2. La població de

cada servei és independent a les altres i per tan, per a cada servei que s'hagi indicat des de les pantalles del programa, i fins que no es superi el temps de simulació estipulat per l'usuari:

1.- Es determinen els paràmetres del servei: número de servidors, taxa d'arribada, paràmetres del temps de servei i distribució estadística.

2.- Es generen totes les arribades possibles dins el temps de simulació establert i es guarden en una llista ordenada per temps d'arribada.

Al final s'obté una llista per servei que conté les arribades per a cada servei, on s'indica el temps d'arribada del client i el temps de servei. Durant la iteració de la simulació aquestes llistes es filtren i s'ordenen segons els temps d'arribada per generar una sola cua definitiva. Això igual que al cas anterior provoca que no sempre el temps de rellotge que marca l'arribada d'un client coincideixi amb l'arribada del client anterior més el temps entre arribades, doncs aquest temps entre arribades fa referència a arribades entre clients del mateix servei.

Iteració:

3.- Es busca d'entre les llistes de cada servei el següent client, que serà el que tingui el temps d'arribada més petit.

4.- Aquest mira l'estat dels servidors, si algun està lliure s'hi col·loca

5.- Si tots els servidors estan ocupats mira quina és la cua amb la longitud més curta i s'hi col·loca.

5.- Es busca el següent client d'entre les llistes d'arribades generades (aquell que tingui el temps d'arribada més petit).

6.- S'actualitzen els estats dels servidors i de la cua.

Al final de la simulació el programa retorna una llista de clients per a cada servidor.

7. DISSENY I FUNCIONAMENT DE LA INTERFÍCIE

La interacció amb l'usuari es duu a terme de forma pràctica i ràpida gràcies a la interfície gràfica creada. Aquesta consta de tres pantalles diferents, que es s'expliquen a continuació per tal de poder fer-ne un bon ús.

7.1 PANTALLA INICI

Aquesta pantalla és la que apareix a l'executar el programa i és comuna per a tots els casos

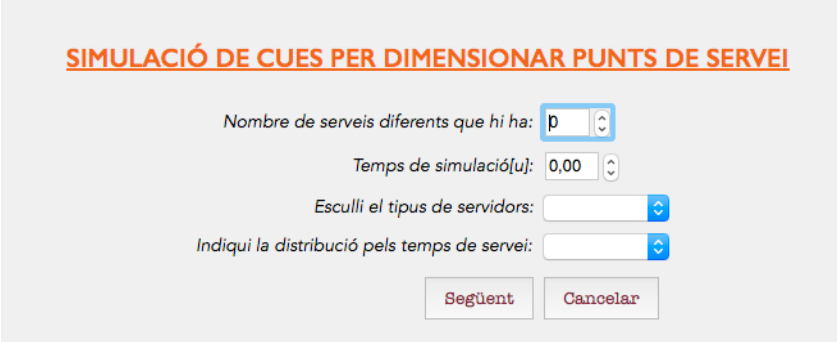


Fig 12. Pantalla d'inici

En aquesta pantalla s'indica:

- Els número de serveis que es volen introduir al sistema que ha de ser major que zero i fins a tres.
- El temps de simulació.
- El tipus de servidor. Com s'ha detallat anteriorment aquests poden atendre un únic servei o ser multiservei.
- La distribució pels temps de servei. Com s'ha explicat es pot escollir entre una distribució exponencial, una normal truncada als positius o una uniforme.

Finalment, el botó *Següent* avança a la següent pantalla i el de *Cancel·lar* tanca el programa.

7.2 PANTALLES INTERMITGES

La segona pantalla que es veurà en cada simulació dependrà dels paràmetres indicats en la primera pantalla. Sigui quin sigui el cas, doncs ens poden sortir 18 pantalles diferents, totes s'organitzen de la mateixa forma variant petites opcions segons el que s'hagi escollit prèviament.

- DIFERÈNCIA ENTRE DISTRIBUCIONS

Si s'escull una distribució uniforme, caldrà introduir per a cada servei el temps mínim i màxim de l'interval de temps de servei.

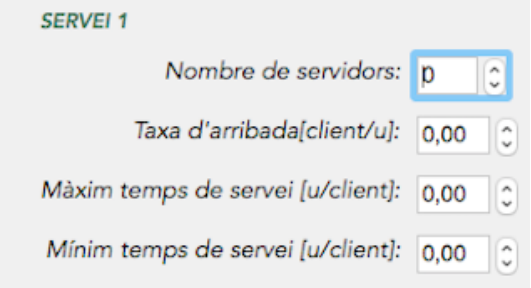


Fig 13. Pantalla amb opció de distribució uniforme pels temps de servei.

Si s'escull una distribució normal caldrà indicar-ne el temps mig i la desviació estàndard.

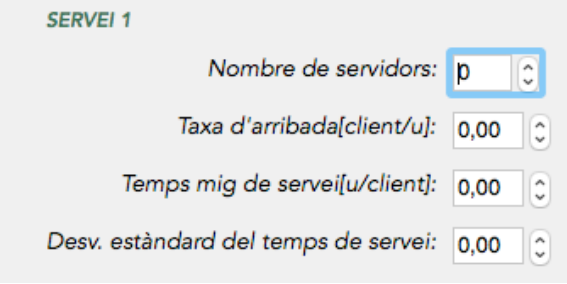


Fig 14. Pantalla amb opció de distribució normal pels temps de servei.

Finalment si s'escull l'opció de distribució exponencial, cal indicar la mitja del temps de servei.

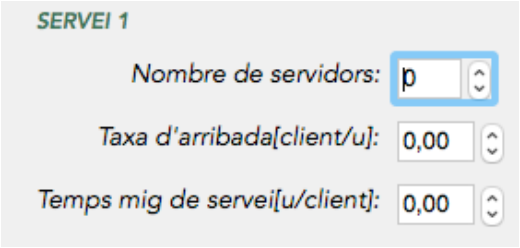


Fig 15. Pantalla amb opció de distribució exponencial pel temps de servei.

- DIFERÈNCIES ENTRE SERVIDORS

La diferència entre escollir servidors que gestionin un únic tipus de servei o servidors multiservei dóna lloc a que calgui indicar el número de servidors per a cada servei o el número de servidors total respectivament.

Per tan, si s'escull un tipus de servidor d'un únic servei, dins de cada servei se'ns demanarà que indiquem el número de servidors, que pot ser diferent entre els diferents serveis.

Fig 16. Pantalla amb l'opció de servidors d'un únic servei.

D'altra banda, l'opció per afegir el número de serveis, podem observar en la fig. 17, que es situa a d'alt de tot junt amb el tipus de cua, ja que el nombre de servidors indicats atendran tots els números de servei que s'hagin definit.

Fig 17. Pantalla amb l'opció de servidors multiservei.

- DIFERÈNCIES EN EL NÚMEROS DE SERVEI

L'única diferència en el número de serveis indicats es que la pantalla ens demanarà les dades per definir cada un dels serveis per separat.

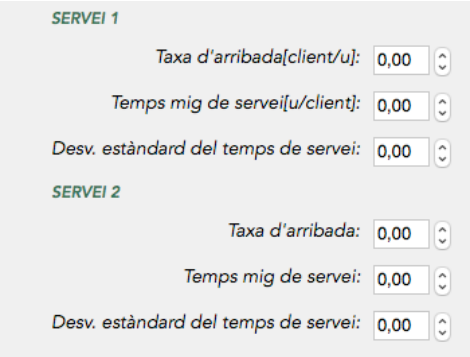


Fig 18. Pantalla amb l'opció de definir dos serveis

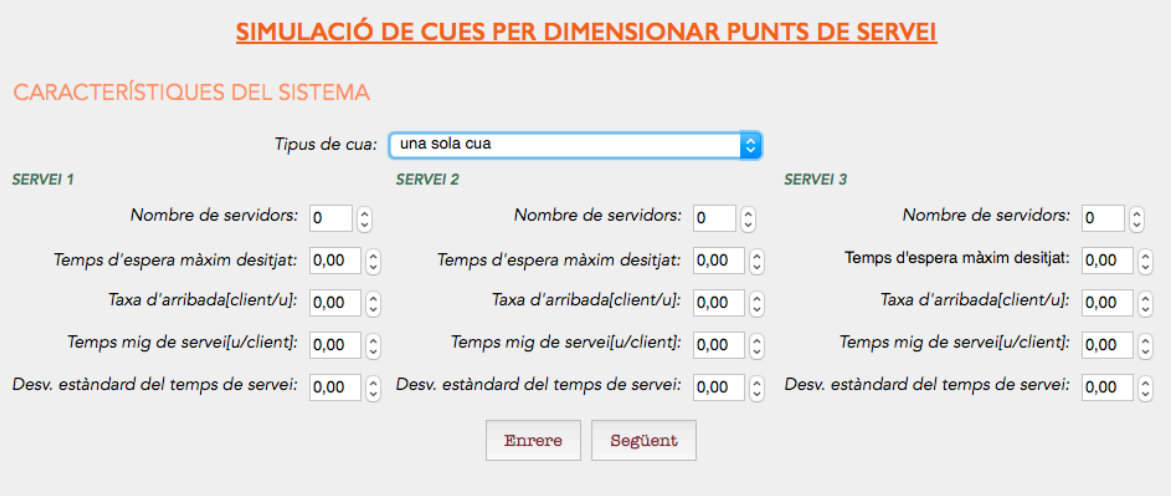


Fig 19. Pantalla amb l'opció d definir tres serveis.

Finalment remarcar que totes les pantalles tenen un botó de *Següent* que executa la simulació i porta a la pantalla de resultats, i un botó de *Enrere* que ens retorna a la pantalla inicial.

- DIFERÈNCIES ENTRE EL TEMPS MÀXIM D'ESPERA

La diferència és si cal determinar un temps màxim per servei, en el cas de servidors independents, o un temps màxim conjunt, en el cas de servidors multiservei. Com es mostra a continuació:

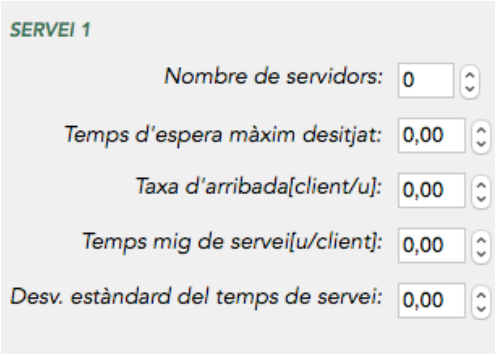


Fig 20. Pantalla amb temps d'espera propi del servei.

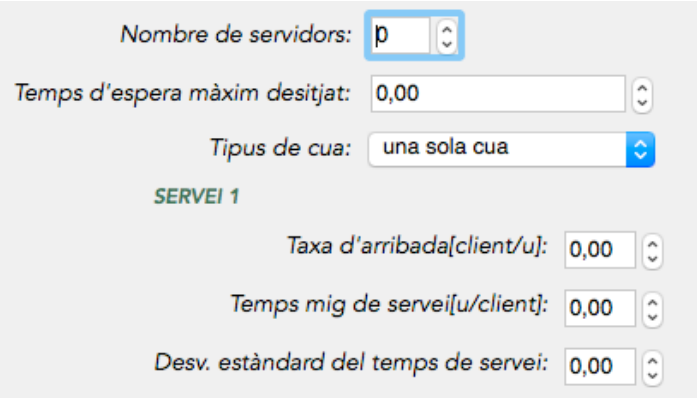


Fig 21. Pantalla amb temps d'espera global per a tots els serveis.

7.3 PANTALLA RESULTATS

L'última pantalla és la que ens dóna accés als resultats. Un primer botó *"Inici"* ens retorna a la pantalla d'inici. Així, si un cop s'obté la taula es vol guardar i realitzar un altra simulació no cal reiniciar el programa. El botó "Generar *Simulació"* ens obra l'arxiu Excel que conté la simulació del sistema introduït, i finalment, el botó *"Generar Solució"* ens obra l'Excel amb una simulació nova feta que indica el nombre mínim de servidors que caldrien per assegurar que ningú sobrepassa el temps màxim d'espera desitjat.

Fig 22. Pantalla final d'accés als resultats.

8. PRESENTACIÓ DE RESULTATS

Donat que el programa pot simular diversos casos, les taules amb les simulacions generades, tot i seguir la mateixa línia, i amb l'objectiu de ser el més clares possibles, varien subtilment entre cada cas.

De totes formes, a part de la informació introduïda per l'usuari, per saber de quina simulació es tracta, sempre es presenta una taula de resultats on s'hi mostra:

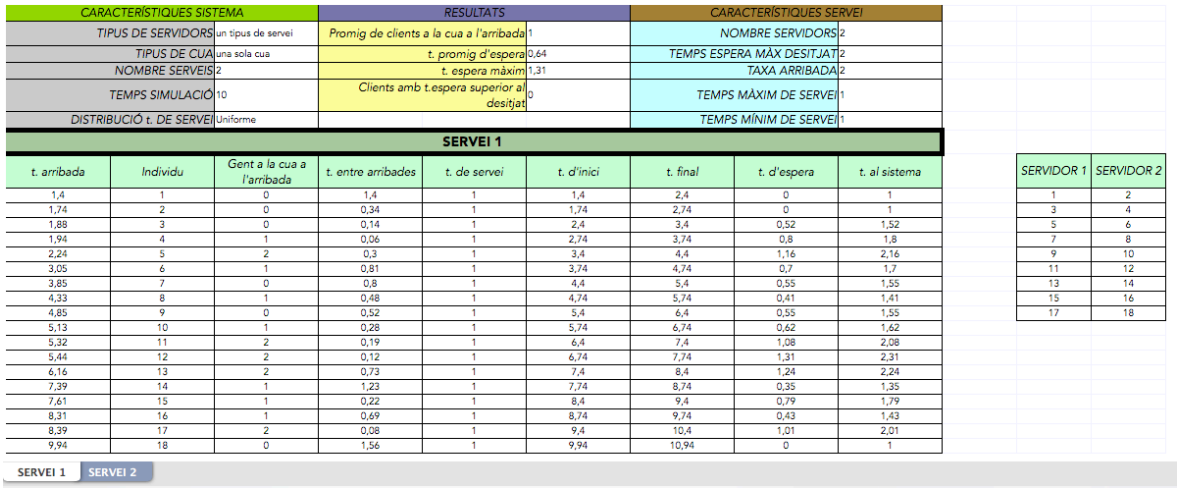
- Als arxius del tipus *Simulació*, s'indica el nombre de servidors que l'usuari ha entrat.
- Als arxius del tipus *Solució,* s'indica el nombre mínim de serveis que cal per tal que ningú s'esperi més del desitjat.

  Com s'ha comentat, cal tenir present que es tracta d'una simulació generada amb variables aleatòries i per tant tot i introduïdes les mateixes característiques el sistema pot diferir entre simulacions. Convé doncs generar varies mostres per avaluar quina solució ens convé.
- El promig de gent a la cua a l'arribada.
- El promig del temps d'espera.
- El màxim temps d'espera. És a dir, el temps del client que està més temps a la cua.
- El nombre de clients que tenen un temps d'espera superior al desitjat.

Evidentment, en els arxius de *Solució*, el màxim temps d'espera no supera el desitjat, i el nombre de clients amb espera major a la desitjada és zero.

A continuació es presenta un exemple per cada un dels quatre casos possibles.

SISTEMA TIPUS 1: Un tipus de servei – Una única cua.



CARACTERÍSTIQUES SISTEMA		RESULTATS		CARACTERÍSTIQUES SERVEI	
TIPUS DE SERVIDORS	un tipus de servei	Promig de clients a la cua a l'arribada	1	NOMBRE SERVIDORS	2
TIPUS DE CUA	una sola cua	t. promig d'espera	0,64	TEMPS ESPERA MÀX DESITJAT	2
NOMBRE SERVEIS	2	t. espera màxim	1,31	TAXA ARRIBADA	2
TEMPS SIMULACIÓ	10	Clients amb t.espera superior al desitjat	0	TEMPS MÀXIM DE SERVEI	1
DISTRIBUCIÓ t. DE SERVEI	Uniforme			TEMPS MÍNIM DE SERVEI	1

SERVEI 1

t. arribada	Individu	Gent a la cua a l'arribada	t. entre arribades	t. de servei	t. d'inici	t. final	t. d'espera	t. al sistema
1,4	1	0	1,4	1	1,4	2,4	0	1
1,74	2	0	0,34	1	1,74	2,74	0	1
1,88	3	0	0,14	1	2,4	3,4	0,52	1,52
1,94	4	1	0,06	1	2,74	3,74	0,8	1,8
2,24	5	2	0,3	1	3,4	4,4	1,16	2,16
3,05	6	1	0,81	1	3,74	4,74	0,7	1,7
3,85	7	0	0,8	1	4,4	5,4	0,55	1,55
4,33	8	1	0,48	1	4,74	5,74	0,41	1,41
4,85	9	0	0,52	1	5,4	6,4	0,55	1,55
5,13	10	1	0,28	1	5,74	6,74	0,62	1,62
5,32	11	2	0,19	1	6,4	7,4	1,08	2,08
5,44	12	2	0,12	1	6,74	7,74	1,31	2,31
6,16	13	2	0,73	1	7,4	8,4	1,24	2,24
7,39	14	1	1,23	1	7,74	8,74	0,35	1,35
7,61	15	1	0,22	1	8,4	9,4	0,79	1,79
8,31	16	1	0,69	1	8,74	9,74	0,43	1,43
8,39	17	2	0,08	1	9,4	10,4	1,01	2,01
9,94	18	0	1,56	1	9,94	10,94	0	1

SERVIDOR 1	SERVIDOR 2
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18

SERVEI 1 SERVEI 2

Fig 23. Pantalla de resultats per al primer tipus i dos serveis.

Com s'observa a la figura 23, cada servei introduït, al ser independent, genera un full nou. Aquest full està distribuït de la següent manera:

A d'alt es poden veure tres rectangles amb informació diferent. El de més a l'esquerra ens diu les característiques del sistema que hem entrat (tipus de cua, de servidor, temps de simulació, nombre de serveis i distribució del temps de servei). El del mig, amb títol de color lila, ens diu els resultats (Promig de gent a la cua a l'arribada, Promig del temps d'espera, Temps màxim d'espera, Gent amb temps d'espera superior al temps màxim). Finalment el resum del servei mostrat al full (nombre de servidors, temps d'espera màxim desitjat pel servei, freqüència d'arribada, i els paràmetres corresponents segons la distribució del temps de servei).

A continuació hi ha el bloc *Cua.* La distribució d'aquest bloc és comú en tots els casos, i cada bloc d'aquests fa referència a una cua generada. Per tant, en aquest cas, al tractar-se d'una simulació del Tipus I, només es genera una cua per servei, i per tant només ens apareixerà un bloc per full. A la cua es mostren totes les dades generades per fer la simulació.

- Rellotge: Marca el temps d'arribada del client i es pot obtenir sumant el temps entre arribades del client amb el temps d'arribada del client anterior. En els casos de servidors multiservei, com s'ha comentat, això és així però cal anar a buscar el client

anterior que venia pel mateix servei, no el directament anterior ja que pot ser client d'un altre servei.

- Individu: Marca l'ordre d'arribada dels clients. En el cas que només hi ha una cua el nombre de l'individu queda ordenat. En el cas que hi hagi una cua per servidor, es pot buscar cada individu en quina cua es situa.

- Temps entre arribades: Temps entre l'arribada anterior (del mateix tipus de servei) i el nou client. Obtingut a partir de la distribució exponencial.

- Temps de servei: Temps que el sistema tardarà en atendre el client. Obtingut a partir de la distribució indicada.

- Temps d'inici: Temps en que el client passarà a ser atès.

- Temps final: Temps d'inici més temps de servei.

- Temps d'espera: La diferència entre el rellotge i el temps d'inici.

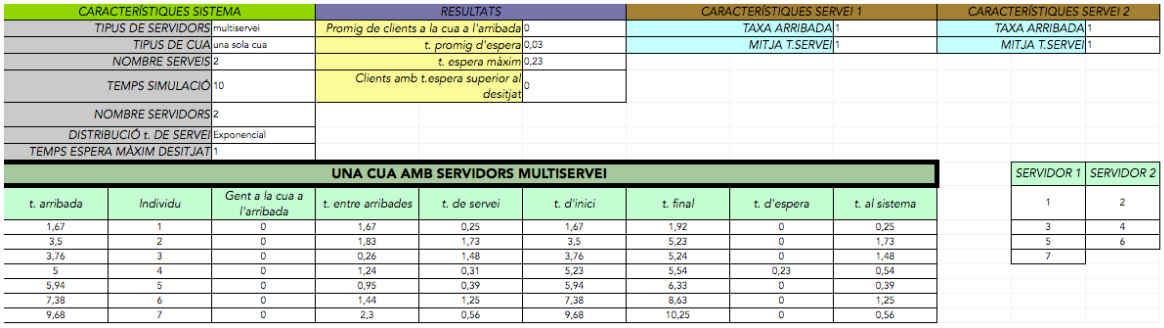
- Temps al sistema: La diferència entre el rellotge i el temps final.

Finalment en aquest tipus d'escenari, al tractar-se d'una sola cua, el programa mostra també en quin servidor s'atén a cada client.

SISTEMA TIPUS 2: Multiservei – Una única cua

El tipus de sistema dos, presenta la distribució següent. A d'alt, els requadres amb informació del sistema, els resultats i els serveis. Aquest cop com els servidors són multiservei, i només es crea una sola cua, es presenta en una mateixa pàgina tota la informació. Així tindrem a d'alt a la dreta tants requadres com serveis haguem introduït.

A continuació és visualitza el bloc de la cua generada i al costat, igual que al cas anterior, una llista on es mostra en quin servidor s'atén a cada individu.



CARACTERÍSTIQUES SISTEMA		RESULTATS		CARACTERÍSTIQUES SERVEI 1		CARACTERÍSTIQUES SERVEI 2	
TIPUS DE SERVIDORS	multiservei	Promig de clients a la cua a l'arribada	0	TAXA ARRIBADA	1	TAXA ARRIBADA	1
TIPUS DE CUA	una sola cua	t. promig d'espera	0,03	MITJA T.SERVEI	1	MITJA T.SERVEI	1
NOMBRE SERVEIS	2	t. espera màxim	0,23				
TEMPS SIMULACIÓ	10	Clients amb t.espera superior al desitjat	0				
NOMBRE SERVIDORS	2						
DISTRIBUCIÓ t. DE SERVEI	Exponencial						
TEMPS ESPERA MÀXIM DESITJAT	1						

UNA CUA AMB SERVIDORS MULTISERVEI

t. arribada	Individu	Gent a la cua a l'arribada	t. entre arribades	t. de servei	t. d'inici	t. final	t. d'espera	t. al sistema
1,67	1	0	1,67	0,25	1,67	1,92	0	0,25
3,5	2	0	1,83	1,73	3,5	5,23	0	1,73
3,76	3	0	0,26	1,48	3,76	5,24	0	1,48
5	4	0	1,24	0,31	5,23	5,54	0,23	0,54
5,94	5	0	0,95	0,39	5,94	6,33	0	0,39
7,38	6	0	1,44	1,25	7,38	8,63	0	1,25
9,68	7	0	2,3	0,56	9,68	10,25	0	0,56

SERVIDOR 1	SERVIDOR 2
1	2
3	4
5	6
7	

Fig 24. Pantalla de resultats per al segon tipus i dos serveis

SISTEMA TIPUS 3: Un tipus de servei – Una cua per servidor

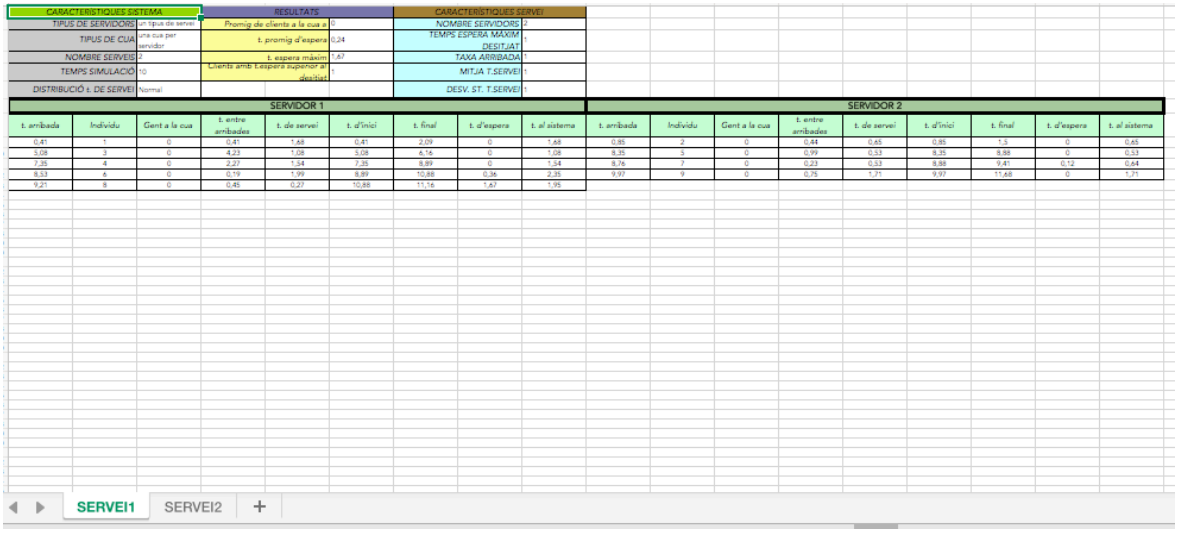
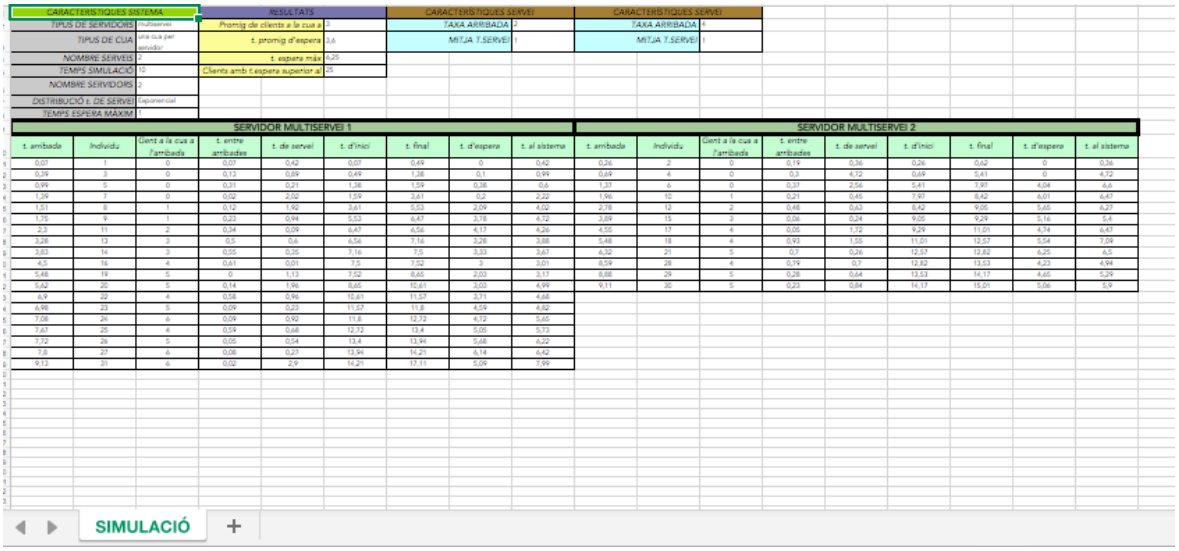


Fig 25. Pantalla de resultats per al segon tipus i dos serveis

En aquest cas, com en el primer, es genera un full per a cada servei. Amb la particularitat que ara tenim una cua per a cada servidor. Així, en cada full se'ns presenta un servei, i epr cada servei, tantes cues com servidors.

SISTEMA TIPUS 4: Multiservei – Una cua per servidor

Finalment, aquest tipus de sistema ens presentarà un únic full, doncs els servidors són multiservei, però a diferència del tipus II, tindrem una cua per a cada servidor. I a d'alt un quadre resum amb els paràmetres de cada servei.



CARACTERÍSTIQUES SISTEMA		RESULTATS		CARACTERÍSTIQUES SERVEI		CARACTERÍSTIQUES SERVEI	
TIPUS DE SERVIDORS	multiservei	Promig de clients a la cua a	4	TAXA ARRIBADA	2	TAXA ARRIBADA	4
TIPUS DE CUA	una cua per servidor	t. promig d'espera	2,6	MITJA T.SERVEI	1	MITJA T.SERVEI	1
NOMBRE SERVEIS	2	t. espera màx	6,25				
TEMPS SIMULACIÓ	10	Clients amb t.espera superior al	25				
NOMBRE SERVIDORS	2						
DISTRIBUCIÓ t. DE SERVEI	Exponencial						
TEMPS ESPERA MÀXIM	1						

SERVIDOR MULTISERVEI 1								
t. arribada	Individu	Client a la cua a l'arribada	t. entre arribades	t. de servei	t. d'inici	t. final	t. d'espera	t. al sistema
0,07	1	0	0,07	0,42	0,07	0,49	0	0,42
0,39	3	0	0,13	0,89	0,49	1,38	0,1	0,99
0,99	5	0	0,31	0,21	1,38	1,59	0,38	0,6
1,39	7	0	0,02	2,02	1,59	3,61	0,2	2,22
1,51	8	1	0,12	1,92	3,61	5,53	2,09	4,02
1,75	9	1	0,23	0,94	5,53	6,47	3,78	4,72
2,3	11	2	0,34	0,09	6,47	6,56	4,17	4,26
3,28	13	3	0,5	0,6	6,56	7,16	3,28	3,88
3,83	14	3	0,55	0,35	7,16	7,5	3,33	3,67
4,5	16	4	0,61	0,01	7,5	7,52	3	3,01
5,48	19	5	0	1,13	7,52	8,65	2,03	3,17
5,62	20	5	0,14	1,96	8,65	10,61	3,03	4,99
6,9	22	4	0,58	0,96	10,61	11,57	3,71	4,68
6,98	23	5	0,09	0,23	11,57	11,8	4,59	4,82
7,08	24	6	0,09	0,92	11,8	12,72	4,72	5,65
7,67	25	4	0,59	0,68	12,72	13,4	5,05	5,73
7,72	26	5	0,05	0,54	13,4	13,94	5,68	6,22
7,8	27	6	0,08	0,27	13,94	14,21	6,14	6,42
9,13	31	6	0,02	2,9	14,21	17,11	5,09	7,98

SERVIDOR MULTISERVEI 2								
t. arribada	Individu	Client a la cua a l'arribada	t. entre arribades	t. de servei	t. d'inici	t. final	t. d'espera	t. al sistema
0,26	2	0	0,19	0,36	0,26	0,62	0	0,36
0,69	4	0	0,3	4,72	0,69	5,41	0	4,72
1,37	6	0	0,37	2,56	5,41	7,97	4,04	6,6
1,96	10	1	0,21	0,45	7,97	8,42	6,01	6,47
2,78	12	2	0,48	0,63	8,42	9,05	5,65	6,27
3,89	15	3	0,06	0,24	9,05	9,29	5,16	5,4
4,55	17	4	0,05	1,72	9,29	11,01	4,74	6,47
5,48	18	4	0,93	1,56	11,01	12,57	5,54	7,09
6,32	21	5	0,7	0,24	12,57	12,82	6,25	6,5
8,59	28	4	0,79	0,7	12,82	13,53	4,23	4,94
8,88	29	5	0,28	0,64	13,53	14,17	4,65	5,29
9,11	30	5	0,23	0,84	14,17	15,01	5,06	5,9

SIMULACIÓ +

Fig 26. Pantalla de resultats per al segon tipus i dos serveis

9. EXEMPLE D'APLICACIÓ

Suposem el cas d'una línia aèria i els seus taulells de facturació, i que de tots els disponibles n'obren 4 per un dels vols.

La classe *Economic* té un temps d'entre 6 i 10 minuts, doncs les dimensions i el pes és més estricte, i cal revisar-ho. I el temps d'espera no és important, doncs a la companyia ja li va bé que esperin, perquè així la pròxima vegada es pensaran en agafar una tarifa més cara així que posarem un màxim de 40 minuts..

La classe *Business* com tenen més preferències no són tan estrictes i el temps de gestió és més ràpid. D'entre uns 3 i 7 minuts. El temps d'espera màxim desitjat per aquesta classe és zero, doncs volem tenir els clients contents perquè segueixin volant amb aquesta tarifa.

Al principi, que hi ha més gent de la classe *Economic* que *Business* perquè saben que hauran de fer cua, es dediquen tres servidors als d'*Economic* i un al de *Business.* Ara bé, què passa quan s'apropa l'hora d'embarcar i arriben els que queden per facturar?

CARACTERÍSTIQUES SISTEMA:

Número de serveis	Economic/Business
Temps de simulació	60 min
Tipus de servidors	Un tipus de servei
Distribució temps de servei	Uniforme
Tipus de cua	Una cua per servidor

Taula 2 Exemple. Característiques del sistema.

CLASSE *ECONOMIC*

Taxa arribada	0,5 persona/minut
Màxim temps de servei	10 minuts
Mínim temps de servei	6 minuts
Temps d'espera desitjat	40 min

Taula 3 Exemple. Característiques del servei 1.

CLASSE *BUSINESS*

Taxa arribada	0,1 persona/minut
Màxim temps de servei	7 minuts
Mínim temps de servei	3 minuts
Temps d'espera màxim desitjat	0 minuts

Taula 4 Exemple. Característiques del servei 2.

Les taules la solució que ens genera el programa es poden veure a les taules de l'annex d'aquesta memòria,

Com se'ns indica, caldrien mínim 2 servidors per la classe *Business*. Ara és qüestió de la companyia decidir si un dels mostradors que atenia la classe *Economic* passa a atendre els *Business* fent les cues pels primers encara més llargues o si s'obra un nou mostrador.

L'opció d'obrir un nou punt de servei, si es té el recurs, sempre hi és, però com tots hem comprovat alguna vegada, no sempre interessa.

10. MILLORES I FUTUR DEL PROJECTE

Finalment, aquest és un programa pensat per a que sigui fàcil d'implementar. I de fet, durant el desenvolupament del projecte se li han afegit moltes millores. Els arxius independents del que es forma el software permeten que la implementació de noves funciones sigui fàcil .

Tot i així sempre es pot millorar, i aquí presento algunes de les següents millores que es podrien incloure:

- En els casos d'escenaris multiservei, es podria indicar per a quin tipus de servei ve cada client.
- Incloure clients V.I.P. que a l'arribar es puguin colar.
- Incloure una taxa d'abandonaments de la cua.
- Que el propi programa generi diferents simulacions per tal de donar un nombre mínim de servidors necessari més fiable.
- Incloure gràfics de probabilitat de gent a la cua a l'arribada.

Aquest són alguns exemples, però segur que a cada usuari se li acudirien de diferents. És per això que el codi del programa pretén ser accessible a tothom així es dóna l'oportunitat de que cadascú el millori o l'adapti a les seves necessitats.

11. VALORACIÓ IMPACTE AMBIENTAL

La valoració mediambiental és important en qualsevol tipus d'activitat i projecte. En aquest cas, i tractant-se del desenvolupament d'un programa informàtic, l'impacte ambiental degut a l'aplicació del projecte es redueix al consum energètic d'un ordinador en funcionament.

Segons la Comissió Europea un 1kWh produeix 0,65kg de CO_2:

Consum de potència d'un ordinador mig	220 W
Hores de funcionament aproximades a l'any (suposant 5h/dia, 5 dies la setmana i 50 setmanes l'any)	1250 h
Consum d'energia	275 kWh
Consum de CO_2	178,75 kg

Taula 5. Impacte ambiental

12. VALORACIÓ IMPACTE ECONÒMIC

A continuació es detallen els costos de realització del projecte amb la finalitat de realitzar una valoració econòmica:

Lectures i cerca d'informació relacionada	18h
Recerca i estudi d'alternatives de software	30h
Disseny de la interfície	15h
Implementació de la interfície	30h
Disseny del software	50h
Implementació del software	130h
Proves pilots i millores	20h
Redacció de la memòria	30h
TOTAL	323h

Taula 6. Comptabilització d'hores

Considerant un sou d'un enginyer junior de 24.000€ l'any, la feina realitzada implicaria un cost de 15€/h. Per tant, el projecte suposaria un cost de 4.845€ per les hores invertides, i aplicant un I.V.A del 21%, el cost final del projecte és de 5.862,45 €.

Conclusions

És obvi que les cues tendeixen a créixer quan els punts de servei no tenen prou capacitat, i és igualment clar, que no sempre es pot fer front a tota la demanda. Doncs com es comprova amb aquest programa les cues són molt aleatòries i en moments d'afluència màxima és fàcil que es generin esperes.

Si es volgués poder fer front en tot moment al paràmetre de temps d'espera màxim desitjat, el sistema quedaria sobredimensionat en franges on l'afluència de gent fos menor, i sinó, es pateix el risc que els temps d'espera siguin excessius.

En tot cas, s'ha dissenyat un programa informàtic potent, que permet a un usuari sense coneixements d'informàtica ni estadística generar simulacions per a diferents casos amb l'objectiu de treure un dimensionament del nombre de servidors adequat al valor d'espera màxim desitjat. És feina de l'usuari del programa però, la de valorar els resultats presentats segons aspectes econòmics o d'imatge de l'empresa, etc... abans de prendre una decisió.

Agraïments

M'agradaria agrair al meu tutor de projecte, Pere G., per la seva comprensió i consell. A en Josep B per la seva paciència i ajuda en conceptes informàtics. I finalment, a l'Ariadna L. i la Mònica B. per les seves idees i temps invertit..

Bibliografia

Referències bibliogràfiques

[1] Kofman, E y Bergero, F. Laboratorio de Sistemas dinámicos y Procesamiento de la Infomración, Universidad Nacional de Rosario, Argentina. Técnicas de Moldeado y Simulación. <http://www.fceia.unr.edu.ar/~kofman/files/charla_unlam_6.pdf>

[2] Ingolfsson, A y Grossman, T. (2013). Catonsville, EEUU. INFORMS Trnasactions on Education. Graphical Spreadsheet Queueing Simulation <http://www.ualberta.ca/~aingolfs/simulation/>

[3] Editorial Logística La. (2015). Alternativas de software libre para simulación en Logística. <https://editorial.logistica.la/2015/07/29/software-libre-scm/>

[4] Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay. (2011). Herramientas para Simulación a Eventos Discretos. <https://www.fing.edu.uy/inco/cursos/simulacion/archivos/clases/clase15bweb.pdf>

[5] Law, A y Kelton, W. (2000). Simulation Modeling & Analysis.McGraw-Hill, tercera edición,(2000)

[6] Universidad de Buenos Aires, Facultad de Ingenería. (2005). Simulacion. http://materias.fi.uba.ar

[7] Barceló, J. (1996). Simulación de Sistemas Discretos (primera edición). Barcelona: Isdefe.

[8] Python Software Foundation. The Python Standard Library. <https://docs.python.org/2/library/random.html>

[9] Apunts d'Optimització i Simulació, Departament D'organització d'Empreses. ETSEIB.

Bibliografia complementària

- Downey, A., How to think like a computer scientist : learning with Python, Green Tea Press, 2002.
- PyQt4 Documentation. <http://pyqt.sourceforge.net/Docs/PyQt4>
- Zetcode PyQt Tutorials <http://zetcode.com>
- Microsoft, Visual Basic for Applications (VBA) <https://msdn.microsoft.com/eses/library/office/ee814737(v=office.14).aspx >
- Bressert, E., SciPy and NumPy: exmplamples to Jumpstart your scientific PYthon programming. Sebastopol, Ca, O'Reilly, 2012.

ANNEX

1.- Taula 1: Solució a l'exemple proposat per a la classe Business

2.- Taula2: Solució a l'exemple proposat per a la classe Economic (1a part)

3.- Taula 3: Solució a l'exemple proposat per a la classe Economic (2a part)

4.- Arxiu del codi del programa: programapral.py

CARACTERÍSTIQUES SISTEMA		RESULTATS		CARACTERÍSTIQUES SERVEI	
TIPUS DE SERVIDORS	un tipus de servei	Promig de clients a la cua	0	NOMBRE SERVIDORS MÍNIM	2
TIPUS DE CUA	una cua per servidor	t. promig d'espera	0	TEMPS ESPERA MÀXIM DESITJAT	0
NOMBRE SERVEIS	2	t. espera màxim	0	TAXA ARRIBADA	0,2
TEMPS SIMULACIÓ	60	Clients amb t.espera superior al desitjat	0	TEMPS MÀXIM DE SERVEI	7
DISTRIBUCIÓ t. DE SERVEI	Uniforme			TEMPS MÍNIM DE SERVEI	3

SERVIDOR 1						
t. arribada	Individu	Gent a la cua	t. entre arribades	t. de servei	t. d'inici	t. final	t. d'espera	t. al sistema
5,45	1	0	5,45	6,72	5,45	12,17	0	6,72
17,57	3	0	8,37	5,93	17,57	23,5	0	5,93
36,62	5	0	18,4	5,97	36,62	42,59	0	5,97

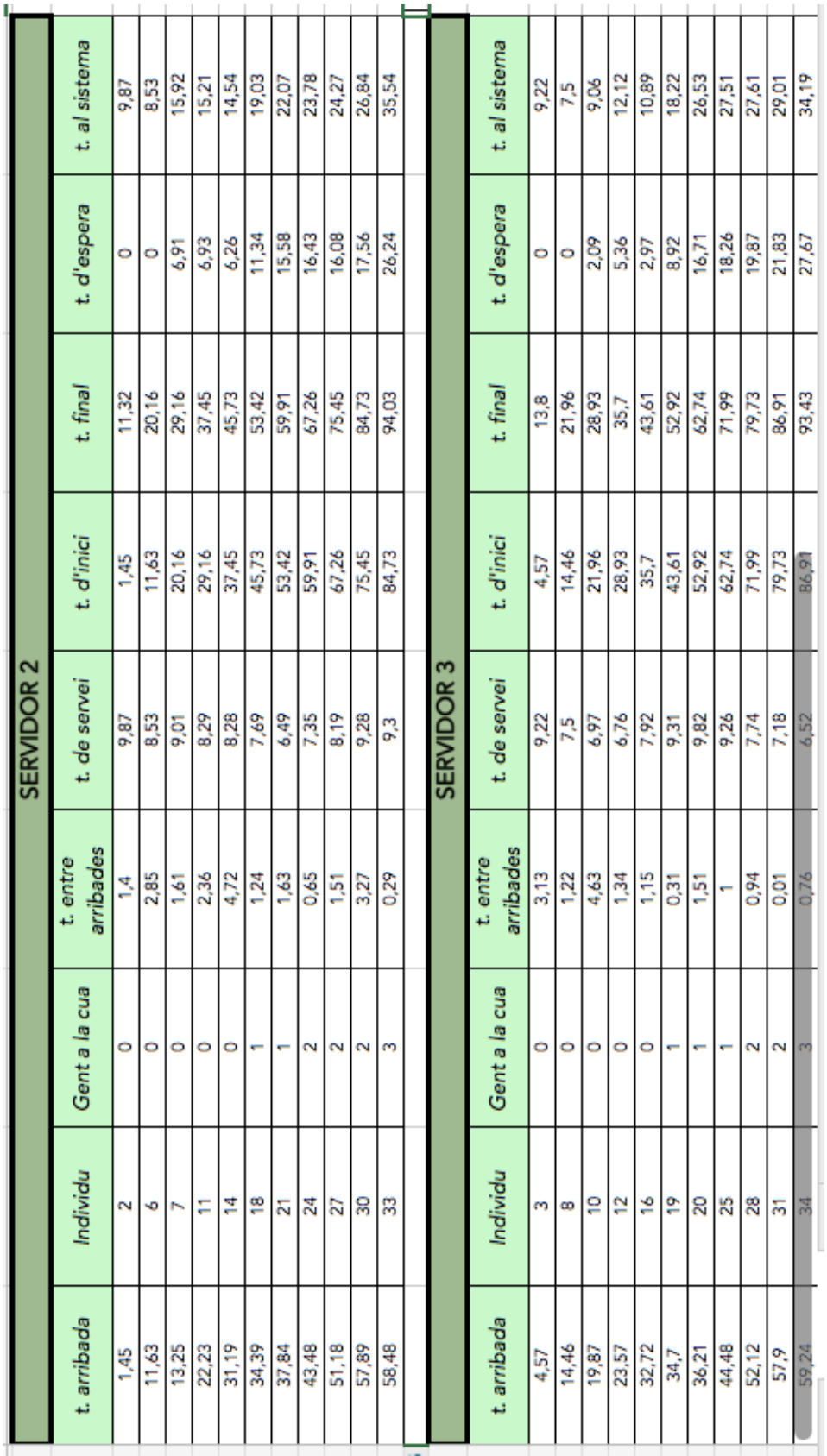
SERVIDOR 2								
t. arribada	Individu	Gent a la cua	t. entre arribades	t. de servei	t. d'inici	t. final	t. d'espera	t. al sistema
9,2	2	0	3,75	4,15	9,2	13,35	0	4,15
18,22	4	0	0,65	6,93	18,22	25,15	0	6,93
38,5	6	0	1,88	4,62	38,5	43,11	0	4,62

TAULA 1. SOLUCIÓ AL EXEMPLE PROPOSAT PER LA CLASSE BUSINESS

CARACTERÍSTIQUES SISTEMA		RESULTATS		CARACTERÍSTIQUES SERVEI	
TIPUS DE SERVIDORS	un tipus de servei	Promig de clients a la cua	2	NOMBRE SERVIDORS MÍNIM	3
TIPUS DE CUA	una cua per servidor	t. promig d'espera	12,61	TEMPS ESPERA MAXIM DESITJAT	40
NOMBRE SERVEIS	1	t. espera màxim	37,76	TAXA ARRIBADA	0,5
TEMPS SIMULACIÓ	60	Clients amb t.espera superior al desitjat	0	TEMPS MÀXIM DE SERVEI	10
DISTRIBUCIÓ t. DE SERVEI	Uniforme			TEMPS MÍNIM DE SERVEI	6

SERVIDOR 1							
t. arribada	Individu	Gent a la cua	t. entre arribades	t. de servei	t. d'inici	t. final	t. d'espera	t. al sistema
0,05	1	0	0,05	7,39	0,05	7,44	0	7,39
5,69	4	0	1,12	7,58	7,44	15,02	1,75	9,33
8,78	5	0	3,09	9,46	15,02	24,48	6,24	15,71
15,24	9	0	0,78	6,79	24,48	31,28	9,25	16,04
26,46	13	0	2,89	7,66	31,28	38,94	4,81	12,47
31,57	15	0	0,38	7,29	38,94	46,22	7,36	14,65
33,15	17	1	0,43	8,05	46,22	54,28	13,07	21,12
40,58	22	1	2,74	9,54	54,28	63,81	13,7	23,24
42,83	23	2	2,25	8,94	63,81	72,76	20,99	29,93
49,68	26	2	5,19	6,84	72,76	79,6	23,08	29,92
54,62	29	2	2,49	9,94	79,6	89,54	24,98	34,92
58,2	32	3	0,3	7,8	89,54	97,34	31,34	39,14
59,59	35	4	0,34	8,89	97,34	106,23	37,76	46,64

TAULA 2. SOLUCIÓ AL EXEMPLE PROPOSAT PER LA CLASSE ECONOMIC (continuació a la següent pàgina)



SERVIDOR 2							
t. arribada	*Individu*	*Gent a la cua*	*t. entre arribades*	*t. de servei*	*t. d'inici*	*t. final*	*t. d'espera*	*t. al sistema*
1,45	2	0	1,4	9,87	1,45	11,32	0	9,87
11,63	6	0	2,85	8,53	11,63	20,16	0	8,53
13,25	7	0	1,61	9,01	20,16	29,16	6,91	15,92
22,23	11	0	2,36	8,29	29,16	37,45	6,93	15,21
31,19	14	0	4,72	8,28	37,45	45,73	6,26	14,54
34,39	18	1	1,24	7,69	45,73	53,42	11,34	19,03
37,84	21	1	1,63	6,49	53,42	59,91	15,58	22,07
43,48	24	2	0,65	7,35	59,91	67,26	16,43	23,78
51,18	27	2	1,51	8,19	67,26	75,45	16,08	24,27
57,89	30	2	3,27	9,28	75,45	84,73	17,56	26,84
58,48	33	3	0,29	9,3	84,73	94,03	26,24	35,54

SERVIDOR 3								
t. arribada	*Individu*	*Gent a la cua*	*t. entre arribades*	*t. de servei*	*t. d'inici*	*t. final*	*t. d'espera*	*t. al sistema*
4,57	3	0	3,13	9,22	4,57	13,8	0	9,22
14,46	8	0	1,22	7,5	14,46	21,96	0	7,5
19,87	10	0	4,63	6,97	21,96	28,93	2,09	9,06
23,57	12	0	1,34	6,76	28,93	35,7	5,36	12,12
32,72	16	0	1,15	7,92	35,7	43,61	2,97	10,89
34,7	19	1	0,31	9,31	43,61	52,92	8,92	18,22
36,21	20	1	1,51	9,82	52,92	62,74	16,71	26,53
44,48	25	1	1	9,26	62,74	71,99	18,26	27,51
52,12	28	2	0,94	7,74	71,99	79,73	19,87	27,61
57,9	31	2	0,01	7,18	79,73	86,91	21,83	29,01
59,24	34	3	0,76	6,52	86,91	93,43	27,67	34,19

TAULA 3. SOLUCIÓ AL EXEMPLE PROPOSAT PER LA CLASSE ECONOMIC.

ARXIU: PROGRAMAPRAL.PY

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from PyQt4 import QtCore, QtGui
from pscreen1 import Ui_screen1
from pscreen2 import Ui_screen2
from pscreen2u import Ui_screen2u
from pscreen2e import Ui_screen2e
from pscreen3 import Ui_screen3
from pscreen3u import Ui_screen3u
from pscreen3e import Ui_screen3e
from pscreen22 import Ui_screen22
from pscreen22u import Ui_screen22u
from pscreen22e import Ui_screen22e
from pscreen23 import Ui_screen23
from pscreen23u import Ui_screen23u
from pscreen23e import Ui_screen23e
from pscreen32 import Ui_screen32
from pscreen32u import Ui_screen32u
from pscreen32e import Ui_screen32e
from pscreen33 import Ui_screen33
from pscreen33u import Ui_screen33u
from pscreen33e import Ui_screen33e
from presultats import Ui_screenresultats
import os
import calculs


class pantalla1(QtGui.QMainWindow, Ui_screen1):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbseguent1.clicked.connect(self.pbseguent1_clicked)
        self.numproductes.setRange(0, 3)
        self.pbcancelar.clicked.connect(QtGui.qApp.quit)
        self.tipusdesimulacio1.addItem("un tipus de servei")
        self.tipusdesimulacio1.addItem("multiservei")
        self.dist.addItem("Normal")
        self.dist.addItem("Exponencial")
        self.dist.addItem("Uniforme")

    def pbseguent1_clicked(self):
        tipusim=str(self.tipusdesimulacio1.currentText())
        np=self.numproductes.value()
        distri=str(self.dist.currentText())
        if distri=='Normal':
            if tipusim=='un tipus de servei':
                if np==1:
                    self.novafinestra = pantalla2(self)
                    self.novafinestra.show()
                    self.close()
                elif np==2:
                    self.novafinestra = pantalla22(self)
                    self.novafinestra.show()
                    self.close()
                elif np==3:
                    self.novafinestra = pantalla23(self)
                    self.novafinestra.show()
                    self.close()
                else:
                    self.statusbar.showMessage("Error: cal seleccionar el nombre de productes que s'ofereixen")
            else:
                if np==1:
                    self.novafinestra=pantalla3(self)
                    self.novafinestra.show()
                    self.close()
```

```
        elif np==2:
            self.novafinestra = pantalla32(self)
            self.novafinestra.show()
            self.close()
        elif np==3:
            self.novafinestra = pantalla33(self)
            self.novafinestra.show()
            self.close()
        else:
            self.statusbar.showMessage("Error: cal seleccionar el nombre de productes que s'ofereixen")
    elif distri=='Exponencial':
        if tipusim=='un tipus de servei':
            if np==1:
                self.novafinestra = pantalla2e(self)
                self.novafinestra.show()
                self.close()
            elif np==2:
                self.novafinestra = pantalla22e(self)
                self.novafinestra.show()
                self.close()
            elif np==3:
                self.novafinestra = pantalla23e(self)
                self.novafinestra.show()
                self.close()
            else:
                self.statusbar.showMessage("Error: cal seleccionar el nombre de productes que s'ofereixen")
        else:
            if np==1:
                self.novafinestra=pantalla3e(self)
                self.novafinestra.show()
                self.close()
            elif np==2:
                self.novafinestra = pantalla32e(self)
                self.novafinestra.show()
                self.close()
            elif np==3:
                self.novafinestra = pantalla33e(self)
                self.novafinestra.show()
                self.close()
            else:
                self.statusbar.showMessage("Error: cal seleccionar el nombre de productes que s'ofereixen")
    elif distri=='Uniforme':
        if tipusim=='un tipus de servei':
            if np==1:
                self.novafinestra = pantalla2u(self)
                self.novafinestra.show()
                self.close()
            elif np==2:
                self.novafinestra = pantalla22u(self)
                self.novafinestra.show()
                self.close()
            elif np==3:
                self.novafinestra = pantalla23u(self)
                self.novafinestra.show()
                self.close()
            else:
                self.statusbar.showMessage("Error: cal seleccionar el nombre de productes que s'ofereixen")
        else:
            if np==1:
                self.novafinestra=pantalla3u(self)
                self.novafinestra.show()
                self.close()
            elif np==2:
                self.novafinestra = pantalla32u(self)
                self.novafinestra.show()
                self.close()
            elif np==3:
```

```
                self.novafinestra = pantalla33u(self)
                self.novafinestra.show()
                self.close()
            else:
                self.statusbar.showMessage("Error: cal seleccionar el nombre de productes que s'ofereixen")
        else:
            self.statusbar.showMessage(u"Error: cal seleccionar el tipus de distribució pels temps de servei")

class pantalla2(QtGui.QMainWindow, Ui_screen2):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()

    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            fa=self.freqarribada1.value()
            fs=self.tserveis1.value()
            des=self.desv1.value()
            tc=self.tipusdecua.currentText()
            ns=self.numservidors1.value()
            te=self.te.value()
            self.serveis1=[ns, te, fa, fs, des]
            self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
            if str(tc)=="una sola cua":
                calculs.primer([self.serveis1], self.sistema)
            else:
                calculs.segon([self.serveis1], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla2u(QtGui.QMainWindow, Ui_screen2u):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()

    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()
```

```
    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            fa=self.freqarribada1.value()
            a=self.tmax.value()
            b=self.tmin.value()
            tc=self.tipusdecua.currentText()
            ns=self.numservidors1.value()
            te=self.te.value()
            self.serveis=[ns, te, fa, a, b]
            self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
            if str(tc)=="una sola cua":
                calculs.primer([self.serveis], self.sistema)
            else:
                calculs.segon([self.serveis], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla2e(QtGui.QMainWindow, Ui_screen2e):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()

    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            fa=self.freqarribada1.value()
            fs=self.tserveis.value()
            tc=self.tipusdecua.currentText()
            ns=self.numservidors1.value()
            te=self.te.value()
            self.serveis=[ns, te, fa, fs]
            self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
            if str(tc)=="una sola cua":
                calculs.primer([self.serveis], self.sistema)
            else:
                calculs.segon([self.serveis], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()


class pantalla22(QtGui.QMainWindow, Ui_screen22):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
```

```
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()

    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        ns2=self.numservidors2.value()
        if ns==0 or ns2==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()
            des=self.desv1.value()
            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
            des2=self.desv2.value()
            te=self.te.value()
            te2=self.te2.value()

            self.servei1=[ns, te, fa, fs, des]
            self.servei2=[ns2, te2, fa2, fs2, des2]
            self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
            if str(tc)=="una sola cua":
                calculs.primer([self.servei1,self.servei2], self.sistema)
            else:
                calculs.segon([self.servei1,self.servei2], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla22u(QtGui.QMainWindow, Ui_screen22u):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()

    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        ns2=self.numservidors2.value()
        if ns==0 or ns2==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            a=self.tmax1.value()
            b=self.tmin1.value()
```

```
        fa2=self.freqarribada2.value()
        a2=self.tmax2.value()
        b2=self.tmin2.value()
        te=self.te.value()
        te2=self.te2.value()

        self.serveil=[ns, te, fa, a, b]
        self.servei2=[ns2, te2, fa2, a2, b2]
        self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
        if str(tc)=="una sola cua":
            calculs.primer([self.serveil,self.servei2], self.sistema)
        else:
            calculs.segon([self.serveil,self.servei2], self.sistema)
        self.novafinestra=presultats(self)
        self.novafinestra.show()
        self.close()

class pantalla22e(QtGui.QMainWindow, Ui_screen22e):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()

    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        ns2=self.numservidors2.value()
        if ns==0 or ns2==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tserveil.value()
            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
            te=self.te.value()
            te2=self.te2.value()


            self.serveil=[ns, te, fa, fs]
            self.servei2=[ns2, te2, fa2, fs2]
            self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
            if str(tc)=="una sola cua":
                calculs.primer([self.serveil,self.servei2], self.sistema)
            else:
                calculs.segon([self.serveil,self.servei2], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla23(QtGui.QMainWindow, Ui_screen23):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
```

```
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()


    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        ns2=self.numservidors2.value()
        ns3=self.numservidors3.value()
        if ns==0 or ns2==0 or ns3==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()
            des=self.desv1.value()
            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
            des2=self.desv2.value()
            fa3=self.freqarribada3.value()
            fs3=self.tservei3.value()
            des3=self.desv3.value()
            te=self.te.value()
            te2=self.te2.value()
            te3=self.te3.value()

            self.servei1=[ns, te, fa, fs, des]
            self.servei2=[ns2, te2, fa2, fs2, des2]
            self.servei3=[ns3, te3, fa3, fs3, des3]
            self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
            if str(tc)=="una sola cua":
                calculs.primer([self.servei1, self.servei2, self.servei3], self.sistema)
            else:
                calculs.segon([self.servei1, self.servei2, self.servei3], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()
class pantalla23u(QtGui.QMainWindow, Ui_screen23u):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()


    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
```

```
        ns=self.numservidors1.value()
        ns2=self.numservidors2.value()
        ns3=self.numservidors3.value()
        if ns==0 or ns2==0 or ns3==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            a=self.tmax1.value()
            b=self.tmin1.value()
            fa2=self.freqarribada2.value()
            a2=self.tmax2.value()
            b2=self.tmin2.value()
            fa3=self.freqarribada3.value()
            a3=self.tmax3.value()
            b3=self.tmin3.value()
            te=self.te.value()
            te2=self.te2.value()
            te3=self.te3.value()
            self.servei1=[ns, te, fa, a, b]
            self.servei2=[ns2, te2, fa2, a2, b2]
            self.servei3=[ns3, te3, fa3, a3, b3]
            self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
            if str(tc)=="una sola cua":
                calculs.primer([self.servei1, self.servei2, self.servei3], self.sistema)
            else:
                calculs.segon([self.servei1, self.servei2, self.servei3], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla23e(QtGui.QMainWindow, Ui_screen23e):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere1_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent2_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()


    def pbenrere1_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent2_clicked(self):
        ns=self.numservidors1.value()
        ns2=self.numservidors2.value()
        ns3=self.numservidors3.value()
        if ns==0 or ns2==0 or ns3==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()
            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
            fa3=self.freqarribada3.value()
            fs3=self.tservei3.value()
            te=self.te.value()
            te2=self.te2.value()
```

```
        te3=self.te3.value()
        self.servei1=[ns, te, fa, fs]
        self.servei2=[ns2, te2, fa2, fs2]
        self.servei3=[ns3, te3, fa3, fs3]

        self.sistema=[tc, self.ts, self.tipusim, self.np, self.dis]
        if str(tc)=="una sola cua":
            calculs.primer([self.servei1, self.servei2, self.servei3], self.sistema)
        else:
            calculs.segon([self.servei1, self.servei2, self.servei3], self.sistema)
        self.novafinestra=presultats(self)
        self.novafinestra.show()
        self.close()


class pantalla3(QtGui.QMainWindow, Ui_screen3):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()


    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()
            des=self.desv1.value()
            te=self.te.value()

            self.servei1=[fa, fs, des]

            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.servei1], self.sistema)
            else:
                calculs.quart([self.servei1], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla3u(QtGui.QMainWindow, Ui_screen3u):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
```

```
        self.dis=Pantalla1.dist.currentText()


    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            a=self.tmax.value()
            b=self.tmin.value()
            te=self.te.value()

            self.serveil=[fa, a, b]

            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.serveil], self.sistema)
            else:
                calculs.quart([self.serveil], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla3e(QtGui.QMainWindow, Ui_screen3e):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()


    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tserveil.value()
            te=self.te.value()

            self.serveil=[fa, fs]

            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.serveil], self.sistema)
            else:
                calculs.quart([self.serveil], self.sistema)
            self.novafinestra=presultats(self)
```

```
        self.novafinestra.show()
        self.close()

class pantalla32(QtGui.QMainWindow, Ui_screen32):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()



    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()
            des=self.desv1.value()
            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
            des2=self.desv2.value()
            te=self.te.value()

            self.servei1=[fa, fs, des]
            self.servei2=[fa2, fs2, des2]

            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.servei1, self.servei2], self.sistema)
            else:
                calculs.quart([self.servei1, self.servei2], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla32u(QtGui.QMainWindow, Ui_screen32u):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()



    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
```

```
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            b=self.tmax1.value()
            a=self.tmin1.value()
            fa2=self.freqarribada2.value()
            a2=self.tmax2.value()
            b2=self.tmin2.value()
            te=self.te.value()

            self.servei1=[fa, a, b]
            self.servei2=[fa2, a2, b2]

            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.servei1, self.servei2], self.sistema)
            else:
                calculs.quart([self.servei1, self.servei2], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla32e(QtGui.QMainWindow, Ui_screen32e):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)
        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")
        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()



    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()
            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
            te=self.te.value()

            self.servei1=[fa, fs]
            self.servei2=[fa2, fs2]

            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.servei1, self.servei2], self.sistema)
            else:
```

```
            calculs.quart([self.servei1, self.servei2], self.sistema)
        self.novafinestra=presultats(self)
        self.novafinestra.show()
        self.close()

class pantalla33(QtGui.QMainWindow, Ui_screen33):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)

        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()


    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()
            des=self.desv1.value()
            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
            des2=self.desv2.value()
            fa3=self.freqarribada3.value()
            fs3=self.tservei3.value()
            des3=self.desv3.value()
            te=self.te.value()
            self.servei1=[fa, fs, des]
            self.servei2=[fa2, fs2, des2]
            self.servei3=[fa3, fs3, des3]
            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.servei1, self.servei2, self.servei3], self.sistema)
            else:
                calculs.quart([self.servei1, self.servei2, self.servei3], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()
class pantalla33u(QtGui.QMainWindow, Ui_screen33u):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)

        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()
```

```
    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tmax1.value()
            des=self.tmin1.value()
            fa2=self.freqarribada2.value()
            fs2=self.tmax2.value()
            des2=self.tmin2.value()
            fa3=self.freqarribada3.value()
            fs3=self.tmax3.value()
            des3=self.tmin3.value()
            te=self.te.value()
            self.servei1=[fa, fs, des]
            self.servei2=[fa2, fs2, des2]
            self.servei3=[fa3, fs3, des3]
            self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
            if str(tc)=="una sola cua":
                calculs.tercer([self.servei1, self.servei2, self.servei3], self.sistema)
            else:
                calculs.quart([self.servei1, self.servei2, self.servei3], self.sistema)
            self.novafinestra=presultats(self)
            self.novafinestra.show()
            self.close()

class pantalla33e(QtGui.QMainWindow, Ui_screen33e):
    def __init__(self, Pantalla1, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pbenrere1.clicked.connect(self.pbenrere_clicked)
        self.pbseguent2.clicked.connect(self.pbseguent_clicked)

        self.tipusdecua.addItem("una sola cua")
        self.tipusdecua.addItem("una cua per servidor")

        self.ts=Pantalla1.tsimulacio.value()
        self.np=Pantalla1.numproductes.value()
        self.tipusim=Pantalla1.tipusdesimulacio1.currentText()
        self.dis=Pantalla1.dist.currentText()


    def pbenrere_clicked(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def pbseguent_clicked(self):
        ns=self.numservidors.value()
        if ns==0:
            self.statusbar.showMessage("Error: cal indicar el nombre de servidors")
        else:
            tc=self.tipusdecua.currentText()
            fa=self.freqarribada1.value()
            fs=self.tservei1.value()

            fa2=self.freqarribada2.value()
            fs2=self.tservei2.value()
```

```
        fa3=self.freqarribada3.value()
        fs3=self.tservei3.value()
        te=self.te.value()

        self.servei1=[fa, fs]
        self.servei2=[fa2, fs2]
        self.servei3=[fa3, fs3]
        self.sistema=[ns, tc, self.ts, self.tipusim, self.np, self.dis, te]
        if str(tc)=="una sola cua":
            calculs.tercer([self.servei1, self.servei2, self.servei3], self.sistema)
        else:
            calculs.quart([self.servei1, self.servei2, self.servei3], self.sistema)
        self.novafinestra=presultats(self)
        self.novafinestra.show()
        self.close()


class presultats(QtGui.QMainWindow, Ui_screenresultats):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self)
        self.setupUi(self)
        self.pushButton.clicked.connect(self.inici)
        self.pbtaula.clicked.connect(self.exportarTaula)
        self.pbsimulacio.clicked.connect(self.obrirsolucio)

    def inici(self):
        self.novafinestra=pantalla1(self)
        self.novafinestra.show()
        self.close()

    def exportarTaula(self):
        try:
            import os
            os.system('open simulacio.xls')
        except:
            self.statusbar.showMessage("Error: obriu el fitxer des de la carpeta del programa")
    def obrirsolucio(self):
        try:
            import os
            os.system('open solucio.xls')
        except:
            self.statusbar.showMessage("Error: obriu el fitxer des de la carpeta del programa")


if __name__=='__main__':
    import sys
    app = QtGui.QApplication(sys.argv)
    window=pantalla1()
    window.show()
    sys.exit(app.exec_())
```

ARXIU: CALCULS.PY

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from PyQt4 import QtCore, QtGui

import random
import string
import math
import numpy as np
import programappl as creartaula
import programasol as crearsolucio



def client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua):
      espera=tinici-rellotge
      tsistema=espera+tservei
      return (round(rellotge,2), ind, gcua, round(tarribada,2), round(tservei,2), round(tinici,2),
round(tfinal,2), round(espera,2), round(tsistema,2))

def unif(a, b):
      return random.uniform(a, b)

def normal(mu, sigma):

   return abs(random.gauss(mu, sigma))

def crear_exp(freq):
      r=random.random()
      x=np.log(r)
      y=-freq*x
      return y

def primer(m, p):
   print m
   np=len(m)
   ts=p[1]
   s={}
   d={}
   dis=p[4]
   dis=str(dis)
   ds={}
   ss={}
   mnsn=[]
   for producte in range(np):
      ns=m[producte][0]
      fa=m[producte][2]
      te=m[producte][1]
      serv=[]
      servs=[]
      for a in range(ns):
            serv.append([])
      rellotge=0
      cua=[]
      tarribada=crear_exp(1/fa)
      if dis=='Normal':
            fs=m[producte][3]
            des=m[producte][4]
            tservei=normal(fs, des)
      elif dis=='Exponencial':
            fs=m[producte][3]
            tservei=crear_exp(fs)
      else:
            a=m[producte][3]
            b=m[producte][4]
```

```
        tservei=unif(a, b)
    rellotge+=tarribada
    ind=1
    while rellotge<ts:
      colocat=False
      j=0
      while j<ns and not colocat:
            if len(serv[j])==0:
              tinici=rellotge
              tfinal=tinici+tservei
              serv[j].append((ind, tfinal))
              colocat=True
            else:
              j+=1
      while not colocat:
            index1=0
            index2=0
            while index1<ns and index2<ns:
              if serv[index1][-1][-1]<serv[index2][-1][-1]:
                petit=serv[index1]
                index2+=1
                pos=index1
              else:
                petit=serv[index2]
                index1+=1
                pos=index2
            tinici=max(serv[pos][-1][-1], rellotge)
            tfinal=tinici+tservei
            serv[pos].append((ind, tfinal))
            colocat=True
      gcua=0
      for gencua in range(len(cua)):
            if cua[gencua][5]>rellotge:
                  gcua+=1
            else:
                  pass
      cua.append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
      tarribada=crear_exp(1/fa)
      if dis=='Normal':
        fs=m[producte][3]
        des=m[producte][4]
        tservei=normal(fs, des)
      elif dis=='Exponencial':
        fs=m[producte][3]
        tservei=crear_exp(fs)
      else:
        a=m[producte][3]
        b=m[producte][4]
        tservei=unif(a, b)
      rellotge+=tarribada
      ind+=1
    d[producte+1]=cua
    s[producte+1]=serv
    tmax=99999
    nsn=0
    print 'ok'
    while (tmax*1.05)>(te+0.000005):
          nsn+=1
          servs=[]
          for a in range(nsn):
                servs.append([])
          fa=m[producte][2]
          te=m[producte][1]
          rellotge=0
          cuas=[]
          tarribada=crear_exp(1/fa)
          if dis=='Normal':
```

```
        fs=m[producte][3]
        des=m[producte][4]
        tservei=normal(fs, des)
    elif dis=='Exponencial':
        fs=m[producte][3]
        tservei=crear_exp(fs)
    else:
        a=m[producte][3]
        b=m[producte][4]
        tservei=unif(a, b)
    rellotge+=tarribada
    ind=1
    while rellotge<ts:
        colocat=False
        j=0
        while j<nsn and not colocat:
            if len(servs[j])==0:
                tinici=rellotge
                tfinal=tinici+tservei
                servs[j].append((ind, tfinal))
                colocat=True
            else:
                j+=1
        while not colocat:
            index1=0
            index2=0
            while index1<nsn and index2<nsn:
                if servs[index1][-1][-1]<servs[index2][-1][-1]:
                    petits=servs[index1]
                    index2+=1
                    pos=index1
                else:
                    petits=servs[index2]
                    index1+=1
                    pos=index2
            tinici=max(servs[pos][-1][-1], rellotge)
            tfinal=tinici+tservei
            servs[pos].append((ind, tfinal))
            colocat=True
        gcua=0
        for gencua in range(len(cuas)):
            if cuas[gencua][5]>rellotge:
                gcua+=1
            else:
                pass
        cuas.append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
        tarribada=crear_exp(1/fa)
        if dis=='Normal':
            fs=m[producte][3]
            des=m[producte][4]
            tservei=normal(fs, des)
        elif dis=='Exponencial':
            fs=m[producte][3]
            tservei=crear_exp(fs)
        else:
            a=m[producte][3]
            b=m[producte][4]
            tservei=unif(a, b)
        rellotge+=tarribada
        ind+=1
    print 'ok2'
    ttt=0
    for usuari in range(len(cuas)):
        if cuas[usuari][7]>=ttt:
            tmax=cuas[usuari][7]
            ttt=tmax
        else:
```

```
                    pass
            print tmax
        print 'ok3'
        mnsn.append(nsn)
        ds[producte+1]=cuas
        ss[producte+1]=servs
        print np

    creartaula.primer(d,p,m,s)
    crearsolucio.primer(ds, p, m, ss, mnsn)

def segon(m, p):
    np=len(m)
    i=0
    ts=p[1]
    ll=[]
    lls=[]
    mnova=[]
    dis=p[4]
    dis=str(dis)
    while i<np:
        ns=m[i][0]
        fa=m[i][2]
        te=m[i][1]
        rellotge=0
        q=[]
        w=[]
        y={}
        k=0
        ind=0
        while k<ns:
            y[k+1]=True
            q.append([])
            w.append([])
            k+=1
        tarribada=crear_exp(1/fa)
        if dis=='Normal':
            fs=m[i][3]
            des=m[i][4]
            tservei=normal(fs, des)
        elif dis=='Exponencial':
            fs=m[i][3]
            tservei=crear_exp(fs)
        else:
            a=m[i][3]
            b=m[i][4]
            tservei=unif(a, b)
        rellotge+=tarribada
        ind=1
        while rellotge<ts:

            for b in range(ns):
                try:
                    if q[b][0]<=rellotge:
                        q[b].pop(0)
                        if len(q[b])==0:
                            y[b+1]=True
                        else:
                            pass
                    else:
                        pass
                except:
                    pass
            j=0
            trobat=False
            while j<ns and not trobat:
                if y[j+1]==True:
```

```
            trobat=True
            y[j+1]=False
            tinici=rellotge
            tfinal=tinici+tservei
            q[j].append(tfinal)
            gcua=0
            for gencua in range(len(w[j])):
              if w[j][gencua][5]>rellotge:
                    gcua+=1
              else:
                    pass
            w[j].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
          else:
            pass
          j+=1
        a=0
        pos=0
        poc=0
        while a<ns and pos<ns and not trobat:
          if len(q[pos])<len(q[a]):
            petita=q[pos]
            a+=1
            poc=pos
          else:
            petita=q[a]
            pos+=1
            poc=a
        if not trobat:
          tinici=max(petita[-1], rellotge)
          tfinal=tinici+tservei
          petita.append(tfinal)
          gcua=0
          for gencua in range(len(w[poc])):
                if w[poc][gencua][5]>rellotge:
                    gcua+=1
                else:
                    pass
          w[poc].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
        tarribada=crear_exp(1/fa)
        if dis=='Normal':
          fs=m[i][3]
          des=m[i][4]
          tservei=normal(fs, des)
        elif dis=='Exponencial':
          fs=m[i][3]
          tservei=crear_exp(fs)
        else:
          a=m[i][3]
          b=m[i][4]
          tservei=unif(a, b)
        rellotge+=tarribada
        ind+=1

      ll.append(w)
      nsn=0
      tmax=99999
      while (tmax*1.05)>(te+0.0000001):
          qs=[]
          ws=[]
          ys={}
          k=0
          ind=0
          rellotge=0
          nsn+=1
          print nsn
          while k<nsn:
              ys[k+1]=True
```

```
        qs.append([])
        ws.append([])
        k+=1
    tarribada=crear_exp(1/fa)
    if dis=='Normal':
        fs=m[i][3]
        des=m[i][4]
        tservei=normal(fs, des)
    elif dis=='Exponencial':
        fs=m[i][3]
        tservei=crear_exp(fs)
    else:
        a=m[i][3]
        b=m[i][4]
        tservei=unif(a, b)
    rellotge+=tarribada
    ind=1
    while rellotge<ts:
        for b in range(nsn):
            try:
                if qs[b][0]<=rellotge:
                    qs[b].pop(0)
                    if len(qs[b])==0:
                        ys[b+1]=True
                    else:
                        pass
                else:
                    pass
            except:
                pass
        j=0
        trobat=False

        while j<nsn and not trobat:

            if ys[j+1]==True:
                trobat=True
                ys[j+1]=False
                tinici=rellotge
                tfinal=tinici+tservei
                qs[j].append(tfinal)
                gcua=0
                for gencua in range(len(ws[j])):
                    if ws[j][gencua][5]>rellotge:
                        gcua+=1
                    else:
                        pass
                ws[j].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
            else:
                pass
            j+=1

        a=0
        pos=0
        poc=0
        while a<nsn and pos<nsn and not trobat:
            if len(qs[pos])<len(qs[a]):
                petitas=qs[pos]
                a+=1
                poc=pos
            else:
                petitas=qs[a]
                pos+=1
                poc=a
            print petitas
        if not trobat:
            tinici=max(petitas[-1], rellotge)
```

```
                    tfinal=tinici+tservei
                    petitas.append(tfinal)
                    gcua=0
                    for gencua in range(len(ws[poc])):
                        if ws[poc][gencua][5]>rellotge:
                            gcua+=1
                        else:
                            pass

                    ws[poc].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
                print 'NOUUUUUUUUUUUUUUUU'
                tarribada=crear_exp(1/fa)
                if dis=='Normal':
                    fs=m[i][3]
                    des=m[i][4]
                    tservei=normal(fs, des)
                elif dis=='Exponencial':
                    fs=m[i][3]
                    tservei=crear_exp(fs)
                else:
                    a=m[i][3]
                    b=m[i][4]
                    tservei=unif(a, b)
                rellotge+=tarribada
                ind+=1
            ttt=0
            print ws
            for service in range(len(ws)):
                tmax
                for usuari in range(len(ws[service])):
                    print ws[service][usuari][7]
                    print ttt
                    if ws[service][usuari][7]>=ttt:
                        tmax=ws[service][usuari][7]
                        ttt=tmax
                    else:
                        pass
            print tmax
        print 'ok3'
        mnova.append(nsn)
        lls.append(ws)
        i+=1
    creartaula.segon(ll, p, m)
    crearsolucio.segon(lls, p, m, mnova)

def tercer(m, p):
    np=len(m)
    ns=p[0]
    ts=p[2]
    q=[]
    dis=p[5]
    dis=str(dis)
    te=p[6]
    for i in range(np):
        fa=m[i][0]
        rellotge=0
        q.append([])
        while rellotge<ts:
            tarribada=crear_exp(1/fa)
            if dis=='Normal':
                fs=m[i][1]
                des=m[i][2]
                tservei=normal(fs, des)
            elif dis=='Exponencial':
                fs=m[i][1]
                tservei=crear_exp(fs)
            else:
```

```
            a=m[i][1]
            b=m[i][2]
            tservei=unif(a, b)
        rellotge+=tarribada
        q[i].append([rellotge,tarribada,tservei])
    mcua=[]
    if np==1:
        mcua=q[0]
    elif np==2:
        p1=0
        p2=0
        while p1<len(q[0]) and p2<len(q[1]):
            if q[0][p1][0]<q[1][p2][0]:
                mcua.append(q[0][p1])
                p1+=1
            else:
                mcua.append(q[1][p2])
                p2+=1
        if p1<len(q[0]):
            while p1<len(q[0][p1]):
                mcua.append(q[0][p1])
                p1+=1
        else:
            while p2<len(q[1]):
                mcua.append(q[1][p2])
                p2+=1
    else:
        p1=0
        p2=0
        p3=0
        while p1<len(q[0]) and p2<len(q[1]) and p3<len(q[2]):
            if q[0][p1][0]<=q[1][p2][0] and q[0][p1][0]<=q[2][p3][0]:
                mcua.append(q[0][p1])
                p1+=1
            elif q[1][p2][0]<=q[0][p1][0] and q[1][p2][0]<=q[2][p3][0]:
                mcua.append(q[1][p2])
                p2+=1
            elif q[2][p3][0]<=q[0][p1][0] and q[2][p3][0]<=q[1][p2][0]:
                mcua.append(q[2][p3])
                p3+=1
            else:
                pass
        if not p1<len(q[0]):
            while p2<len(q[1]) and p3<len(q[2]):
                if q[1][p2][0]<q[2][p3][0]:
                    mcua.append(q[1][p2])
                    p2+=1
                else:
                    mcua.append(q[2][p3])
                    p3+=1
            if p2<len(q[1]):
                while p3<len(q[2]):
                    mcua.append(q[2][p3])
                    p3+=1
            elif p3<len(q[2]):
                while p2<len(q[1]):
                    mcua.append(q[1][p2])
                    p2+=1
            else:
                pass
        elif not p2<len(q[1]):
            while p1<len(q[0]) and p3<len(q[2]):
                if q[0][p1][0]<q[2][p3][0]:
                    mcua.append(q[0][p1])
                    p1+=1
                else:
                    mcua.append(q[2][p3])
```

```
            p3+=1
        if p1<len(q[0]):
            while p3<len(q[2]):
                mcua.append(q[2][p3])
                p3+=1
        elif p3<len(q[2]):
            while p1<len(q[0]):
                mcua.append(q[0][p1])
                p1+=1
        else:
            pass
    elif not p3<len(q[2]):
        while p1<len(q[0]) and p2<len(q[1]):
            if q[0][p1][0]<q[1][p2][0]:
                mcua.append(q[0][p1])
                p1+=1
            else:
                mcua.append(q[1][p2])
                p2+=1
        if p1<len(q[0]):
            while p2<len(q[1]):
                mcua.append(q[1][p2])
                p2+=1
        elif p2<len(q[1]):
            while p1<len(q[0]):
                mcua.append(q[0][p1])
                p1+=1
        else:
            pass
serv=[]
for servidor in range(ns):
    serv.append([])
cua=[]

ind=1
rellotge=0
tarribada=mcua[0][1]
tservei=mcua[0][2]
rellotge=mcua[0][0]
ccc=1
while ccc<len(mcua) and rellotge<ts:
    colocat=False
    j=0
    while j<ns and not colocat:
        if  len(serv[j])==0:
            tinici=rellotge
            tfinal=tinici+tservei
            serv[j].append((ind, tfinal))
            colocat=True
        else:
            j+=1
    while not colocat:
        index1=0
        index2=0
        while index1<ns and index2<ns:
            if serv[index1][-1][-1]<serv[index2][-1][-1]:
                index2+=1
                pos=index1
            else:
                index1+=1
                pos=index2
        tinici=max(serv[pos][-1][-1], rellotge)
        tfinal=tinici+tservei
        serv[pos].append((ind, tfinal))
        colocat=True
    gcua=0
    for gencua in range(len(cua)):
```

```
            if cua[gencua][5]>rellotge:
                gcua+=1
            else:
                pass
        cua.append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
        tarribada=mcua[ccc][1]
        tservei=mcua[ccc][2]
        rellotge=mcua[ccc][2]
        ind+=1
        ccc+=1
    print 'ooook1'
    nsn=0
    tmax=99999
    while (tmax*1.05)>(te+0.000001):
        print 'nouuu'
        nsn+=1
        cuas=[]
        servs=[]
        ccc=1
        ind=1
        rellotge=0
        tarribada=mcua[0][1]
        tservei=mcua[0][2]
        rellotge=mcua[0][0]
        for servidor in range(nsn):
            servs.append([])
        while ccc<len(mcua) and rellotge<ts:
            colocat=False
            j=0
            while j<nsn and not colocat:
                    if  len(servs[j])==0:
                        tinici=rellotge
                        tfinal=tinici+tservei
                        servs[j].append((ind, tfinal))
                        colocat=True
                    else:
                        j+=1
            while not colocat:
                index1=0
                index2=0
                while index1<nsn and index2<nsn:
                    if servs[index1][-1][-1]<servs[index2][-1][-1]:
                        index2+=1
                        pos=index1
                    else:
                        index1+=1
                        pos=index2
                tinici=max(servs[pos][-1][-1], rellotge)
                tfinal=tinici+tservei
                servs[pos].append((ind, tfinal))
                colocat=True
            gcua=0
            for gencua in range(len(cuas)):
                if cuas[gencua][5]>rellotge:
                    gcua+=1
                else:
                    pass
            cuas.append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
            tarribada=mcua[ccc][1]
            tservei=mcua[ccc][2]
            rellotge=mcua[ccc][0]
            ind+=1
            ccc+=1
            print 'okeiiiii'
        print 'ok3'
        ttt=0
        for usuari in range(len(cuas)):
```

```
            if cuas[usuari][7]>=ttt:
                tmax=cuas[usuari][7]
                ttt=tmax
            else:
                pass
        print tmax
    pnova=nsn
    creartaula.tercer(cua, p, m, serv)
    crearsolucio.tercer(cuas, p, m, servs, pnova)


def quart(m, p):
    np=len(m)
    ns=p[0]
    ts=p[2]
    i=0
    q=[]
    te=p[6]
    for i in range(np):
        fa=m[i][0]
        dis=p[5]
        dis=str(dis)
        rellotge=0
        q.append([])
        while rellotge<ts:
            tarribada=crear_exp(1/fa)
            if dis=='Normal':
                fs=m[i][1]
                des=m[i][2]
                tservei=normal(fs, des)
            elif dis=='Exponencial':
                fs=m[i][1]
                tservei=crear_exp(fs)
            else:
                a=m[i][1]
                b=m[i][2]
                tservei=unif(a, b)
            rellotge+=tarribada
            q[i].append([rellotge,tarribada,tservei])
    mcua=[]
    if np==1:
        mcua=q[0]
    elif np==2:
        p1=0
        p2=0
        while p1<len(q[0]) and p2<len(q[1]):
            if q[0][p1][0]<q[1][p2][0]:
                mcua.append(q[0][p1])
                p1+=1
            else:
                mcua.append(q[1][p2])
                p2+=1
        if p1<len(q[0]):
            while p1<len(q[0][p1]):
                mcua.append(q[0][p1])
                p1+=1
        else:
            while p2<len(q[1]):
                mcua.append(q[1][p2])
                p2+=1
    else:
        p1=0
        p2=0
        p3=0
        while p1<len(q[0]) and p2<len(q[1]) and p3<len(q[2]):
            if q[0][p1][0]<=q[1][p2][0] and q[0][p1][0]<=q[2][p3][0]:
                mcua.append(q[0][p1])
```

```
                p1+=1
            elif q[1][p2][0]<=q[0][p1][0] and q[1][p2][0]<=q[2][p3][0]:
                mcua.append(q[1][p2])
                p2+=1
            elif q[2][p3][0]<=q[0][p1][0] and q[2][p3][0]<=q[1][p2][0]:
                mcua.append(q[2][p3])
                p3+=1
            else:
                pass
        if not p1<len(q[0]):
            while p2<len(q[1]) and p3<len(q[2]):
                if q[1][p2][0]<q[2][p3][0]:
                    mcua.append(q[1][p2])
                    p2+=1
                else:
                    mcua.append(q[2][p3])
                    p3+=1
            if p2<len(q[1]):
                while p3<len(q[2]):
                    mcua.append(q[2][p3])
                    p3+=1
            elif p3<len(q[2]):
                while p2<len(q[1]):
                    mcua.append(q[1][p2])
                    p2+=1
            else:
                    pass
        elif not p2<len(q[1]):
            while p1<len(q[0]) and p3<len(q[2]):
                if q[0][p1][0]<q[2][p3][0]:
                    mcua.append(q[0][p1])
                    p1+=1
                else:
                    mcua.append(q[2][p3])
                    p3+=1
            if p1<len(q[0]):
                while p3<len(q[2]):
                    mcua.append(q[2][p3])
                    p3+=1
            elif p3<len(q[2]):
                while p1<len(q[0]):
                    mcua.append(q[0][p1])
                    p1+=1
            else:
                    pass
        elif not p3<len(q[2]):
            while p1<len(q[0]) and p2<len(q[1]):
                if q[0][p1][0]<q[1][p2][0]:
                    mcua.append(q[0][p1])
                    p1+=1
                else:
                    mcua.append(q[1][p2])
                    p2+=1
            if p1<len(q[0]):
                while p2<len(q[1]):
                    mcua.append(q[1][p2])
                    p2+=1
            elif p2<len(q[1]):
                while p1<len(q[0]):
                    mcua.append(q[0][p1])
                    p1+=1
            else:
                    pass
    print mcua
    h=[]
    w=[]
    y={}
```

```
gcua=0
ind=0
k=0
while k<ns:
    y[k+1]=True
    h.append([])
    w.append([])
    k+=1
rellotge=0
tarribada=mcua[0][1]
tservei=mcua[0][2]
rellotge+=tarribada
ind=1
iii=1
while iii<len(mcua):
    if rellotge<ts:
        for b in range(ns):
            try:
                if h[b][0]<=rellotge:
                    h[b].pop(0)
                    if len(h[b])==0:
                        y[b+1]=True
                    else:
                        pass
                else:
                    pass
            except:
                pass
        j=0
        trobat=False
        while j<ns and not trobat:
            name=j+1
            if y[name]==True:
                trobat=True
                y[name]=False
                tinici=rellotge
                tfinal=tinici+tservei
                h[j].append(tfinal)
                gcua=0
                for gencua in range(len(w[j])):
                    if w[j][gencua][5]>rellotge:
                        gcua+=1
                    else:
                        pass
                w[j].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
            else:
                pass
            j+=1
        a=0
        pos=0
        poc=0
        while a<ns and pos<ns and not trobat:
            if len(h[pos])<len(h[a]):
                petita=h[pos]
                a+=1
                poc=pos
            else:
                petita=h[a]
                pos+=1
                poc=a
        if not trobat:
            tinici=max(petita[-1], rellotge)
            tfinal=tinici+tservei
            petita.append(tfinal)
            gcua=0
            for gencua in range(len(w[poc])):
                if w[poc][gencua][5]>rellotge:
```

```
                    gcua+=1
                else:
                    pass
            w[poc].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
        tarribada=mcua[iii][1]
        tservei=mcua[iii][2]
        rellotge+=tarribada
        ind+=1
        iii+=1

nsn=1
tmax=99999
while (tmax*1.05)>(te+0.00001):
    nsn+=1
    hs=[]
    ws=[]
    ys={}
    gcua=0
    k=0
    while k<nsn:
        ys[k+1]=True
        hs.append([])
        ws.append([])
        k+=1
    rellotge=0
    tarribada=mcua[0][1]
    tservei=mcua[0][2]
    rellotge=mcua[0][0]
    ind=1
    iii=1
    while iii<len(mcua) and rellotge<ts:
        for b in range(nsn):
            try:
                if hs[b][0]<=rellotge:
                    hs[b].pop(0)
                    if len(hs[b])==0:
                        ys[b+1]=True
                    else:
                        pass
                else:
                    pass
            except:
                pass
        j=0
        trobat=False
        while j<nsn and not trobat:
            name=j+1
            if ys[name]==True:
                trobat=True
                ys[name]=False
                tinici=rellotge
                tfinal=tinici+tservei
                hs[j].append(tfinal)
                gcua=0
                for gencua in range(len(ws[j])):
                    if ws[j][gencua][5]>rellotge:
                        gcua+=1
                    else:
                        pass
                ws[j].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
            else:
                pass
            j+=1
        a=0
        pos=0
        poc=0
        while a<nsn and pos<nsn and not trobat:
```

```
                if len(hs[pos])<len(hs[a]):
                    petitas=hs[pos]
                    a+=1
                    poc=pos
                else:
                    petitas=hs[a]
                    pos+=1
                    poc=a
            if not trobat:
                tinici=max(petitas[-1], rellotge)
                tfinal=tinici+tservei
                petitas.append(tfinal)
                gcua=0
                for gencua in range(len(ws[poc])):
                    if ws[poc][gencua][5]>rellotge:
                        gcua+=1
                    else:
                        pass
                ws[poc].append(client(ind, rellotge, tarribada, tservei, tinici, tfinal, gcua))
            tarribada=mcua[iii][1]
            tservei=mcua[iii][2]
            rellotge=mcua[iii][0]
            ind+=1
            iii+=1
        ttt=0
        for service in range(len(ws)):
            for usuari in range(len(ws[service])):
                if ws[service][usuari][7]>=ttt:
                    tmax=ws[service][usuari][7]
                    ttt=tmax
        nsn+=1
    pnova=nsn
    creartaula.quart(w, p, m)
    crearsolucio.quart(ws, p, m, pnova)
if __name__=='__main__':
    mmc()
```